Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Detection of fake profiles in Online Social Networks (OSNs)

MASTER DEGREE: Master's degree in Applied Telecommunications and Engineering Management (MASTEAM)

AUTHOR: Beatriche Gencheva

ADVISOR: Olga Leon Abarca

DATE: 10 October 2018

Title: Detection of fake profiles in Online Social Networks (OSNs)

Author: Beatriche Gencheva

Advisor: Olga Leon Abarca

Date: 10 October 2018

Abstract

Nowadays the communication between people is highly influenced by the Online Social Networks (OSNs). Immense number of personal, professional and political thoughts are shared online every single day. Thus, OSNs are attractive for cyber criminals who are trying to exploit their weaknesses and vulnerabilities. Fake accounts on OSNs have become a basic threat used in different online attacks. And even if some of these attacks are harmless like generating fake accounts for “likes” on Facebook, followers on Twitter and views on YouTube, other attacks are more serious and can be dangerous online. Influence on trending topics, spread spam advertisements and false political content are just some of the examples how attackers are able to wreak havoc online by using fake profiles.

With the increasing number of security and privacy threats, some of the OSNs have adopted security measures to stop the mass creation of the fake accounts. However, those measures are often ineffective by the many tools available on the underground marketplaces that allow people to cheaply acquire fake accounts.

In this regard, this thesis aims to detect the fake profiles on a very popular OSN, Twitter, with the help of machine learning algorithms. The first key contribution is the research on the appropriate machine learning techniques. Support Vector Machine, Random Forest and k-Nearest Neighbour were the three supervised learning algorithms. In addition to them, one clustering algorithm was tested, namely k-Means. The next contribution is the acquisition of labelled data related to real and fake profiles in Twitter for the training phase. After analysing the behaviour of the users and their tweets activities, an extensive dataset of 12 features was created. The named Fake profile´s detection dataset plays key role in distinguishing fake accounts among real ones and it is applied on the machine learning algorithms. Analysis of the results has been performed in five different scenarios. The classifiers achieve accuracy score of around 92% for separating the fake profiles from the real ones and the clustering algorithm is able to detect all fake profiles. Finally, for testing purposes were tested some of the followers of Donald Trump with the already trained models.

Acknowledgements

I would like to express my sincere gratitude to my advisor Olga Leon Abarca for giving me the opportunity to work on this really interesting topic. Moreover, I thank her for all the useful comments, remarks, and engagement through the whole process of this master thesis.

I would also like to thank all my professors, teachers, and coaches through the years who made me the motivated, not-giving up person, who always wants to improve and give his best. These qualities were highly necessary during the time spent working on the thesis.

Also I thank to my colleges from the MASTEAM for the productive atmosphere during the last year, for their kindness and loving friendship, for the support, help and ideas throughout this thesis.

I thank to all my loved friends and relatives who have supported me and believed in me throughout the entire process. Special gratitude to my aunt for being my closest person in Barcelona.

Last but not least, I would like to thank my mother, Eleonora Gencheva, and my father, Miroslav Genchev for EVERYTHING. They are my heroes, my inspiration and motivation. I will be grateful forever for having you.
This accomplishment would not have been possible without any of them.

Thank you.

Beatriche Gencheva

CONTENTS

ACRONYMS

API	Application Programming Interface
CV	Cross-validation
kNN	k-Nearest-Neighbour
MI	Mutual Information
OSNs	Online Social Networks
PCA	Principal Component Analysis
PCC	Pearson correlation coefficient
RF	Random Forest
RL	Reinforcement Learning
RBF	Radial Basis Function
RFE	Recursive Feature Elimination
SVM	Support Vector Machine
TV	Television
URL	Uniform Resource Locator

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

OSNs have become the preferred communication method among a diverse set of users including: individual people, companies, and families divided all over the world. There, users are meeting and interacting in Internet through different OSNs and are spending significant amount of time storing and sharing huge amount of personal information. In OSNs, the audience size commanded by an organization or an individual, the so called followers, are measure of how popular that entity is. This measure has important economic and/or political implications [1]. In addition to the rapid growth in technology, this leads to online impersonation, fake profiles creation, security issues and privacy threats [2]. All this fake activities involve mass creation of fake accounts for effectively carrying out online attacks on the OSNs. Some of these attacks are innocuous like generating fake accounts for “likes” on Facebook, followers on Twitter and views on YouTube, but other attacks are more dangerous and can have serious consequences like forcing influence on trending topics, spread spam advertisements and false political content [3]. In addition, over the years fake accounts have been constantly evolving over the years in order to avoid their detection. Thus, it is important to develop techniques for detecting fake accounts, keeping into account their near-real behaviour. In response to this requirement, the goal of the thesis was to develop machine learning based approach to detect fake profiles based on user profile activities and interaction with other users and to evaluate its performance.

Of the different OSNs, Twitter has become popular with users such as young adults, governments, commercial enterprises, and politicians as a way of immediately connecting with their audience and directly conveying their message. Originally started as a personal microblogging site, Twitter [4] has been transformed by common use to an information publishing venue. Statistics reported about a billion of Twitter subscribers, with 335 million monthly active users [5]. Popular public characters, such as actors and singers, as well as radio, TV, and newspapers use Twitter as a new media channel. Globally, Twitter is one of the most popular and widely used OSN and this is the reason why the work on this thesis is concentrated specifically on detecting the Twitter fake profiles addressing the problem by the means of machine learning classification and clustering.

The activities of the users were characterized through an exhaustive feature set covering different aspects of their messages, known as tweets, such as their total number, likes, usage of URL and hashtags, mentioning other users and the user activities like how many accounts is the user following and how many accounts are following him. In total the future set contained 12 features. The dataset with the Twitter profiles was available for research purposes [1] and was containing 1950 real users and 3351 fake users in total. However, for having balance between the classes for the thesis are used 1481 real profiles and 1337 fake profiles. The feature based dataset was examined with three feature selection algorithms – Principal Component Analysis (PCA), Mutual Information (MI) and Recursive Feature Elimination (RFE). It is interesting that the three different approaches ranked as most important different features. The total number of statuses, the average of retweet statuses and the average

hashtags used per status were the best ranked features for PCA, MI and RFE respectively.

Using machine learning approaches is possible to classify if a given profile is legitimate or not. The used classifiers are Support Vector Machine (SVM), Random Forest (RF) and k-Nearest Neighbours (kNN) which are very much efficient in detecting the fake accounts and separate real profiles from the fake profiles with achieving accuracy of 92% with the best hyperparameters of the classifiers.

In the evaluation part of the thesis, the three classifiers were trained with different scaling of the features and the results shows that Random Forest shows the best performance without scaling the features to any different range than the original one. The SVM classifier has the highest scores when all the features are normalized equally in the range between -1 and 1. On the other hand, kNN shows its best results when to the features are assign weights, according to the MI factor.

In additional scenario, the thesis proves that the distribution between the classes in a dataset can lead to high changes in the performance of the machine learning methods. Although it is preferable to have equal number of real and fake users in the dataset, in real life this is almost impossible to be achieved. The reason is that the real profiles are much more than the fake profiles, having a rate of 5% from the total amount of accounts in Twitter [3].

One clustering algorithm, k-Means is tested as well and the achieved score is outstanding, being able to separate all the real accounts form the fake accounts.

As a final step, in the thesis are evaluated 805 followers of Donald Trump. The reason is that OSNs are already increasing the economic and political impact. The obtained results show that between 39% and 46% of this followers are fake. After further analysis of the data was established that the analysed accounts were created 2 months before they were crawled and most of them had not have any activity yet.

The remainder of the document is organized as follows. The first chapter presents the positive aspects of the OSNs together with the negative consequences they can bring with. It gives information about the most popular OSNs nowadays and specifically the social network observed in the thesis, Twitter. In addition, there is explained what are fake profiles and their possible threats to the real users. The next chapter 2 explains some of main machine learning techniques and their working principle. In chapter 3 is defined the dataset used for learning the machine learning algorithms together with the main feature selecting methods. In Chapter 4 is where the algorithm for detecting fake profiles is presented and tested in five different scenarios. Chapter 5 is a case study where part of the Donald Trumps´s followers are analyzed for being real or fake. Finally, the main conclusions and future work of the thesis are presented.

CHAPTER 1. ONLINE SOCIAL NETWORKS (OSNs)

The OSNs are providing the opportunity to people to form new connections and to stay in touch with their relatives and friends through the Internet. Social networking is considered as one of the most successful innovations in the Internet and has grown tremendously through the last twenty years [6].

As a consequence, currently are existing more than 200 well-known social networking services, and thousands of not-so-popular ones [7]. Usually, they are built around some topic or area of interest for the users such as job hunting, online dating, sharing pictures or just connecting friends.

Some of these OSNs have become extremely popular and 6 virtual communities are with more than 1 billion active users (Table 1.1) [8].

Table 1.1 OSNs with more than 1 billion active users

Rank	Name	Active user accounts	Country of origin
1	Facebook	2.23 billion	United States
2	YouTube	1.9 billion	United States
3	WhatsApp	1.5 billion	United States
4	Messenger	1.3 billion	United States
5	Instagram	1 billion	United States
6	WeChat	1 billion	China

Although the number of active users is the usual metric to determine the popularity of the social networks, it is preferable to use alternative criteria because the number of accounts may be artificially inflated. An alternative metric is the traffic generated by OSNs websites, as they are usually associated with the services they offer to their users. These are the top 5 Social Networks using these criteria according to [9] for May 2018.

Table 1.2 Top 5 OSNs according to the generated traffic

Rank	Name	Estimated Unique Monthly Visitors
1	Facebook	1.5 billion
2	YouTube	1.499 billion
3	Twitter	400 million
4	Instagram	275 million
5	LinkedIn	250 million

It is clearly that the first place in popularity by both traffic and number of active users is for Facebook and it is the leading social network in 152 out of 167 countries analysed (91% of the planet). In [10] are analysed the runner-ups and „what would the world be like without Facebook?” The next figure is the map showing the second ranked social networks in 57 nations.

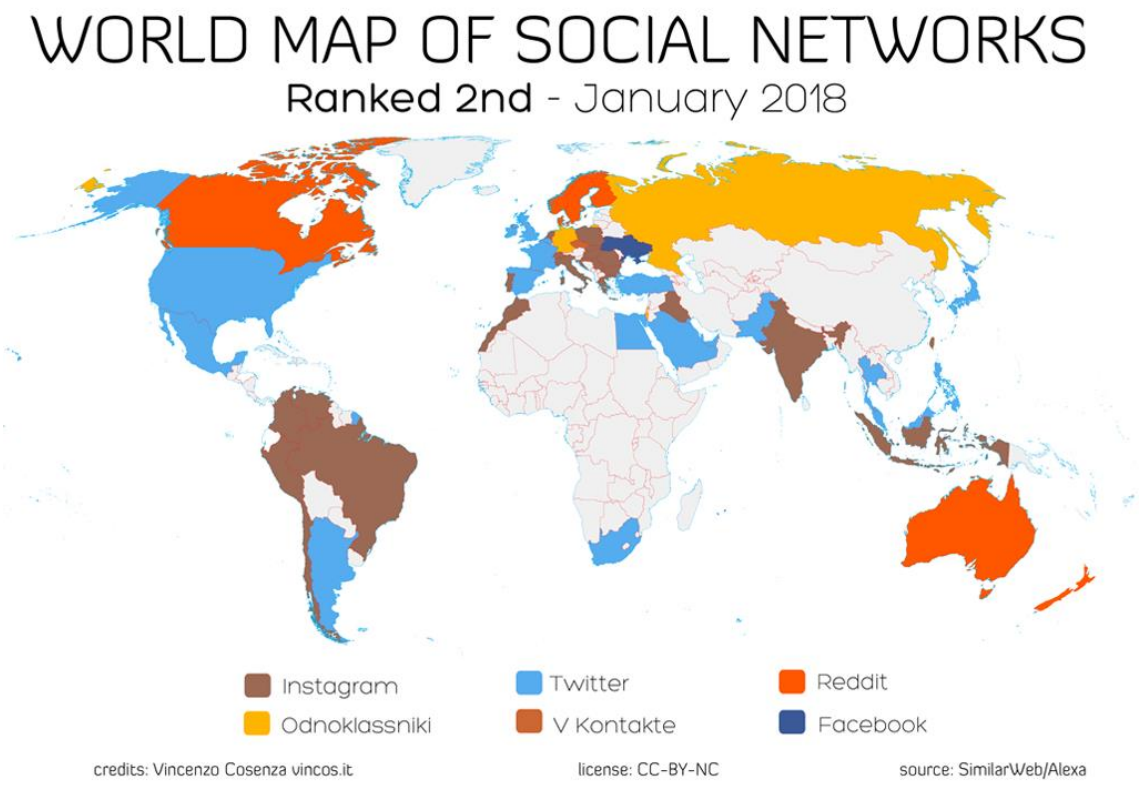

Fig. 1.1 World Map of social networks - Ranked 2nd [10]

Instagram is the runner-up social network in 23 countries. Even though in 2017 Twitter had won the leadership in only 8 countries, currently Twitter has the leader position in 22 nations. This is making it the second fastest growing up OSN in the last year after Facebook.

Twitter´s simplicity is its main attraction along with its immediacy in breaking news about new events. The main reason people use Twitter is because of its interesting content, which is naturally bound to the limitation up to 280 characters per message. A prove is that 500 million of messages are sent every day [11].

All in all, the number of Internet users is 4,038,106,737 and significant amount of them are using OSNs [5]. Before determining one of the major OSNs´ threats for this users, Twitter is described further as the social network used in this thesis.

1.1 Twitter

Twitter [4] is a free social networking and micro-blogging service that allows registered users to send and read status messages known as tweets. Tweets are text-based messages up to 280 characters stored on their author's profile page and sent to other users, known as followers, who have subscribed to them. The sender can restrict access to the status of his or her messages to a circle of friends or allow them to have access to them. When a given user starts following another user, the latter is saved in the list of friends of the first one. If

the user is not registered in the platform, he can only read the tweets, but not write them.

Being designed as a microblogging service showing all their data openly, in Twitter anyone can browse any user´s profile and check its tweets and relationships unless the user requested this data to be private. However, the percentage of private accounts is a minority.

When writing the tweet, it is possible to mention users by including @receiver in the body of the message. In this case, receivers will see the messages even if they are not friends or followers of the sender. Moreover, users can send private messages, known as direct messages, to any of their followers which are not public.

Users usually tag their tweets by the use of #tags inside their tweets. The hashtag is likely the most popular means of categorizing content on social media. It makes the content discoverable and allows to find relevant content from other people and businesses. The hashtag also allows to connect with and engage other social media users based on a common theme or interest. According to the number of tweets using common tags, Twitter creates a ranking known as trending topics in real time showing the most popular tags. It is very popular for users to resend tweets they find especially interesting. Usually they just resend the tweet as they get it, or add some minor additional information. This is known as retweeting. Retweeting helps speeding up the creation of trending topics, all in all making Twitter incredible quick for viral spreading of information.

According to the data available, on December 2017, a total of 1.3 billion accounts have been created. From them 335 million are the monthly active users and approximately 500 million tweets are send each day [5].

Twitter has become popular with users such as young adults, governments, commercial enterprises, and politicians as a way of instantaneously connecting with their audience and directly conveying their message. The success of Twitter (and the other OSNs) as a platform for large scale communication and the expansion of efforts to mine their data for new and novel applications related to public health, economic development, scientific dissemination etc., critically hinges on the authenticity of their user database [12].

Such a flexibility and spread of use have made Twitter the ideal arena for generation of anomalous accounts, that behave in unusual ways. How exactly is explained next.

1.2 Fake profile

In the recent past, media have started reporting that the accounts of politicians, celebrities, and popular brands featured a suspicious rise of followers [13]. The so called fake followers correspond to Twitter accounts specifically exploited to increase the number of followers of a target account. As an example of how important this manipulation is, according to The Washington Post [3] during the 2016 presidential campaign, troll factory was able to use some of America’s

most prominent technology platforms to deceive voters on a mass scale to exacerbate social and political tensions. In addition, during the 2012 US election campaign, the Twitter account of challenger Romney experienced a sudden jump in the number of followers. The great majority of them has been later claimed to be fake [14]. Moreover, having high number of followers in the social medias is from significant meaning for a lot of art professions, like actors, models, musicians where getting a particular job could be not successful if the person is not famous enough online.

Initially, acquiring fake followers could seem a practice limited to increase someone´s confidence, but it could be harmless practice as well. However, artificially inflating the number of followers can also be finalized to make an account more trustworthy and influential, in order to stand from the crowd and to attract other genuine followers.

Taking advantage of the highly dependence of the people on OSNs, an entire industry of black market services has emerged which offers fake accounts for sale. This is exactly how the fake profiles in this thesis were defined (Chapter 3.1 Data acquisition). The increasing number of fake profiles in the last years and the previous mentioned reasons make Twitter to start suspending fake accounts like never before and even putting the company's user growth at risk they removed more than 70 million accounts in May and June this year [3]. After that Twitter was able to announce that fewer than 5% of its active users are fake or involved in spam, and that fewer than 8.5% use automation tools that characterize the accounts as bots. It is important to define that a fake account can also be one that engages in malicious behaviour and is operated by a real person. However, this behaviour is not the case observed in the master thesis.

Moving more aggressively against suspicious accounts will help the platform better to protect users from manipulation and abuse. Therefore, detecting those accounts is necessary nowadays and the goal of this thesis is to overcome current limitations in their characterization. Using machine learning algorithms this is possible to be achieved automatically. What are the different techniques in general and more specifically which algorithms were used in the master thesis, what are their working principles, follow in the next chapter.

CHAPTER 2. MACHINE LEARNING TECHNIQUES

Machine learning is the art of making a computer do useful things without explicitly coding it. More specifically, it is the acquisition of new knowledge through artificial systems. The computer independently generates knowledge from experience and can independently find solutions to new, unknown for it problems, just like every human being. To do this, computer program analyses examples and uses self-learning algorithms to identify patterns in the data. The goal is to intelligently link the data together, recognize possible relationships, make conclusions, and predict [15].

Machine learning is also said to be a subset of artificial intelligence [16]. It enables IT systems to identify patterns and develop solutions based on existing databases and algorithms. Hence, algorithms play a central role in machine learning. They are responsible for recognizing patterns and generating solutions and can be divided into different learning categories. A distinction can be made between supervised learning, in which each example is assigned to a label, the unsupervised case - with data that has not been classified or categorized, and reinforcement learning, where qualitative feedback (interpretable as a reward or punishment) is used for optimization.

Although, for this thesis, both supervised and unsupervised learning has been used for detecting the fake profiles, in the next subchapters, all three techniques will be described further. In addition, there is an explanation of the classification and clustering methods used for that approach.

2.1 Supervised learning

The main goal of supervised learning is by learning a model from training data that has already been labelled to create predictions about unseen, future data. [17] This learning technique is separated in two main subcategories.

The first one is a supervised learning task with discrete class labels, also known as classification. Its objective is to recognize patterns. When the outcome signal is, instead of labels, continuous values, it refers to the second subcategory of supervised learning knows as regression. In this case, it is spoken about function approximation. On the figure below is illustrated the process of making predictions about the future with supervised learning.

The main goal of classification is to predict the categorical class labels of new instances based on previous examinations. Classes are also known as targets, labels or categories. Classification predictive modelling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

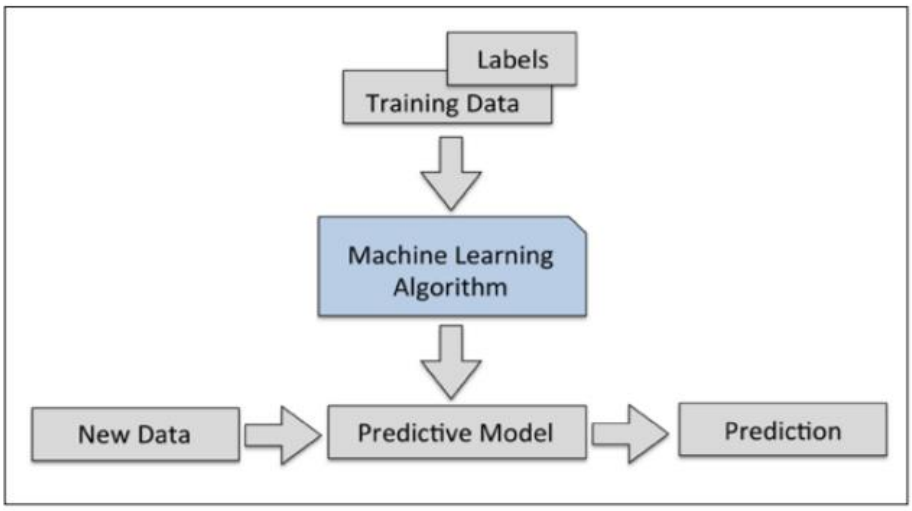

Fig. 2.1 Supervised learning [17]

The detection of fake profiles represents a typical example of a binary classification task, where the machine learning algorithm learns a set of rules in order to distinguish between two possible classes: real and fake profiles. However the set of class labels does not need to be only binary. The predictive model learned by a supervised learning algorithm can assign any class label that was presented in the training dataset to a new, unlabelled sample. When the classes are more than two, it is called multi-class classification [17].

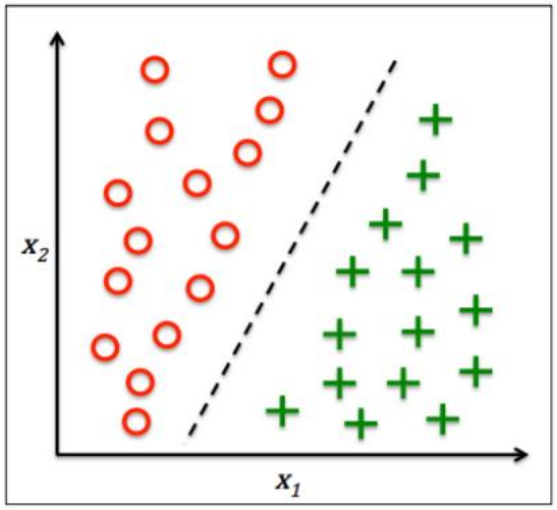

Fig. 2.2 Classification [17]

In the figure above is illustrated an example of binary classification task with in total 30 training samples, separated in labelled as negative and positive class, respectively, circles and plus signs. In this case, the data is two-dimensional and each sample has two values, x_1 and x_2, associated. A rule is the decision boundary represented as a black dashed line in the figure. A supervised machine learning algorithm can learn this rule in order to separate the classes and to classify a new data into each of the two categories given its x_1 and $x_2$values [17].

There are various classification algorithms which can be applied to almost any data problem. Accuracy, training time, linearity, number of parameters and features are some of the points that have to be considered when choosing the best classifier for each specific use case. In this thesis three of the most famous and used algorithms are chosen - Support Vector Machines, Random Forest and k-Nearest-Neighbour. What is their functionality and the advantages compared to the other algorithms is explained below.

2.1.1 Support Vector Machines

Support Vector Machine (SVM) is powerful and widely used machine learning algorithm. Although this supervised learning method can be used for both pattern recognition and function approximation problems, mainly it is used for classification tasks.

SVM is based on hyperplanes that define specific boundaries. A hyperplane is a separation line between set of objects which belongs to different classes [2]. The optimization objective is to maximize the margin which is defined as the distance between the separating hyperplane and the training samples that are closest to this hyperplane [18]. Those training samples are the called support vectors (Fig 2.3).

When speaking of SVM, a data point is viewed as a p-dimensional vector, with p equal to the number of classes. Such points has to be separated with (p-1)-dimensional hyperplane. There are many hyperplanes that might classify the data. The most reasonable choice for finding the best hyperplane is the one that represents the largest separation between the two classes. When maximizing the margin, the decision boundaries tend to have lower generalization error. On the other hand, models with small margins are easier to be overfitted [17].

In the following figure is given an example of binary classification problem with three possible hyperplanes. H1 does not separate the classes at all. H2 does, but only with a small margin. H3 is the optimal one, because it separates both classes with the maximum margin.

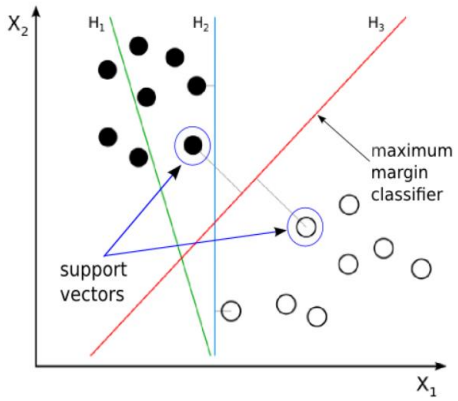

Fig. 2.3 Support Vector Machine [62]

After the explanation of how exactly SVM works, the advantages and disadvantages follow. SVM is memory efficient, because it needs only the

support vectors to classify new data samples [19]. It looks very intuitive as well, however, the mathematical approach behind is very complicated. Furthermore, in some cases the data cannot be separated by linear approach and non-linear SVM has to be used, which may be quite expensive in terms of calculation.

It is effective in high dimensional spaces and when the number of dimensions is greater than the number of samples. However it does not perform well, when the dataset is large, which cause higher training time. Moreover the classes are overlapping when the data set has more noise than usual.

As mentioned before, SVM can be applied for linearly separable data sets as well as non-linearly separable data sets. Those binary sets consist of the ones with fake profile information and the ones with information about the real profiles.

2.1.2 Random Forest

Random Forest (RF) is the second supervised learning algorithm used in the thesis for detection of fake profiles in online social networks. RF is a tree-based method. As such, it is known as really effective and useful method, capable to produce both reliable and understandable results, on mostly any kind of data. The reasons are its good classification performance, scalability, and ease of use.

Random forest is considered as an ensemble of decision trees. By ensemble learning weak learners are integrated with strong learners. This means that the model is at the same time robust, but also has a better generalization error and is less exposed to overfitting [17].

For better understanding of the RF algorithm is necessary to explain what the main idea behind decision trees is. Depending on the features in each dataset, the decision tree model learns a series of questions to figure out the class labels of the instances. What makes this model successful is that it is non-parametric and it can handle heterogeneous data (ordered or categorical variables, or a mix of both). Furthermore decision trees fundamentally implement feature selection, making them at least to some extent robust to irrelevant or noisy variables and are robust to outliers or errors in labels [39].

How RF differ from decision trees and what are the steps it follows is summarized below [17].

1. Randomly choose n samples from the training set with replacement.
2. Grow a decision tree from the n sample. At each node:
    1. Randomly select d features without replacement.
    2. Split the node using the feature that provides the best split according to the objective function, for instance, by maximizing the information gain.
3. Repeat the steps 1 to 2 k times.
4. Aggregate the prediction by each tree to assign the class label by majority vote.

In step 2, when training the individual decision trees: instead of evaluating all features to determine the best split at each node, we only consider a random subset of those. Decision trees are more interpretable, but with RF the tuning is not so important since it is robust to noise from the individual decision trees. A hyperparameter that is important to be tuned is the number of trees and typically, the larger the number of trees, the better the performance of the random forest classifier. However increasing number of trees will cost increased computational cost. In chapter four DETECTION OF FAKE PROFILES IN OSNs BASED ON FEATURE SET, tuning is explained more in details.

Another advantage of RF above decision trees is that RF adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

A reason why RF is one of the preferable machine learning algorithms is that it can be used for both regression and classification tasks. In addition, it is easy to view the relative importance it assigns to the input features.

RF is also considered as a very handy and easy to use algorithm. Firstly, because in most of the use cases with the default hyperparameters the algorithm shows good prediction results. And secondly, if there is a need of tuning the hyperparameters are not a lot and are straightforward to understand. Another positive aspect that RF has is that overfitting will not happen easy, because with enough trees in the forest, it is impossible to overfit the model. One of the biggest problems in machine learning is overfitting and an explanation follows in the fourth chapter of this thesis [20].

As already mentioned, larger number of trees leads to computational cost which makes the algorithm slow and ineffective for real-time predictions. In general, tree-based algorithms are fast to train, but quite slow to create predictions once they are trained. However, in most real-world applications the random forest algorithm is fast enough [20].

2.1.3 k-Nearest Neighbours

The last supervised learning algorithm used in the thesis is the k-nearest neighbour classifier (KNN), which is fundamentally different from the other previously mentioned learning algorithms. This algorithm is one of the most widely used for classification problems since it is simple and easy to implement [17].

KNN is instance-based which means that the algorithm doesn’t explicitly learn a model, but instead it memorizes the training instances which are subsequently used as “knowledge” for the prediction phase [21]. When to the algorithm is given a lot of training data samples along with their features and labels, it stores all of it in the memory, or, rather plots the data in an n-dimensional space. If to a test sample has to be predicted the classification label, the algorithm plots that sample in the same n-dimensional space as the training data. Then, it searches

for its k nearest neighbours based on distance measurement from the training samples. It inspects all the training data each time when it needs to predict a test sample's classification label.

Taking this in account, the minimal training phase of KNN comes both at a memory cost, since a potentially huge data set has to be stored, as well as a computational cost during test time since classifying a given observation requires an examination of the whole data set [21].

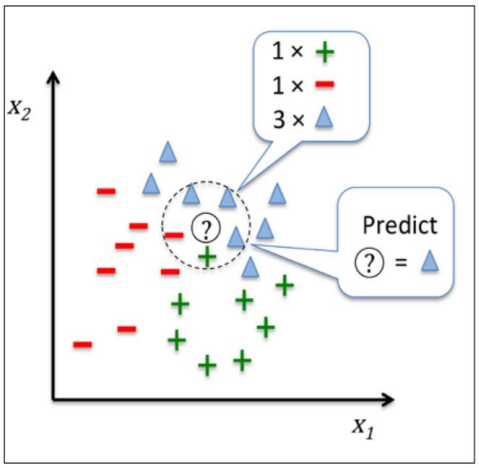

Fig. 2.4 k-Nearest-Neighbour algorithm [17]

In Fig. 2.4 is demonstrated an example how a new "unseen" data point **(?)** is assigned to a classification label based on majority voting. It is clear that two of the nearest neighbours, one plus and one minus, have less majority vote than the three triangles.

Similarity is defined according to a distance metric between two data points. It is important to select the right distance metric, based and dependent on the type of data that has to be processed [22]. The real data set can consist of categorical, numerical or mixed type (both numerical and categorical) of attributes, based on which distance measures may change. In [22], the authors showed that using different distance functions the classification accuracy of the k-NN classifier changes.

Therefore, tuning the hyper parameters is really important for the optimization of the performance of the classifiers and is explained in more detail for each of the three algorithms in chapter 4.3 Parameters.

2.2 Reinforcement learning

Reinforcement Learning (RL) is an area of machine learning, where the learning is achieved when an agent interacts with its environment to achieve a goal. This is often used in situations where a system should learn how to behave optimally within a situation based on positive or negative feedback on a taken action. The system is informed for the situation on the basis of certain input parameters along with a reward of the score [23].

After understanding the main idea of how the algorithm is working it is worth mentioning what are the positive and negative aspects of RL. The first advantage is that there is a balance between trying something already proved in the past and trying new things to reach further improvement. This makes the algorithm more reliable to try new actions or classifications in an incremental format .That is why it can discover new insights and ways of increasing the predictive power. A potential disadvantage could be that is not possible to incorporate explicit rules later on. In addition, a lot of data inputs may be necessary for the machine to receive the proper feedback. On the other hand, RL is more difficult to implement and requires much expertise compared to the other two machine learning techniques [24].

2.3 Unsupervised learning

As mentioned in the previous sections, some networks do not get explicit specifications for the output from the training examples, but only a real-valued signal that can mean 'punishment' or 'reward'. These nets use their weights to avoid 'punishments' in the future.

But when a network does not even receive a scoring signal in response to its outputs, it can still learn something useful and the method is called unsupervised learning. Those networks examine a given dataset to discover possible connections and similarities in the input set.

Learning is often only possible through a description that contains the relevant properties of the inputs. However, it is often not known exactly which properties are important for a successful learning process, and a request to a learning system with disordered or inadequate input can lead to costly calculations or even complete fail.

A technique of unsupervised learning is dimensionality reduction. Most of the times the datasets are composed of high number of features, which means respectively high dimensionality. This could lead to limit the machine learning algorithms in terms of storage space and the computational performance. Moreover, reducing dimensions simplifies large data sets without losing important information, removes noise from the data and makes the data more compact while decreasing the dimensions and in the same time retains the most relevant information [17].

Another main category of techniques for unsupervised learning is finding subgroups with clustering. Since this method is used for the detection of fake profiles in OSNs, it is described further in the next section.

2.2.1 Cluster analysis

The cluster analysis is a classic representative in the field of unsupervised learning. It is a group-building data analysis technique that assigns objects to clusters. The objects assigned to one cluster should be as homogeneous as possible, whereas the objects assigned to different clusters should not share the same degree of similarity.

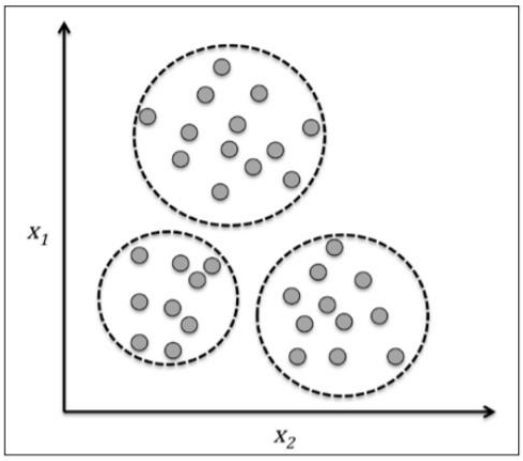

Fig. 2.5 Clustering [17]

The above figure gives an example how, based on the similarity of the two features $x_1$and x_2, three clusters are generated to organise the unlabeled data [17].

2.2.1.1 K-means

K-means is one of the most popular and widely used clustering algorithms. As already mentioned clustering is a technique that finds groups of similar objects, which are more related to each other than to objects in other groups.

This algorithm is easy to implement and also computationally very efficient compared to other clustering algorithms. There are three categories of clustering – prototype-based, hierarchical and density-based. K-means belongs to the first category, namely prototype-based clustering. The other two categories and more clustering algorithms are explained in [17]. In prototype-based clustering each cluster is represented by a prototype, which in the case of similar points of continuous features is the centroid. K-means is very good at identifying clusters of spherical shape. However one of the drawbacks of this clustering algorithm is that the number of clusters k has to be specified in advance. An inappropriate choice for k can result in poor performance of the algorithm. However, in the case of detecting fake profiles this is not a drawback since the number of clusters is clear – one for the real users and one for the fake users. An advantage is that k-means clustering can be applied to data in higher dimensions.

The goal is to group the samples based on their feature similarities and the algorithm can be summarized by the following three steps [25]:

- Initialisation – K centroids are generated at random as initial cluster centres.
- Assignment – Each sample is assigned to the nearest centroid.
- Update – The centre of the samples that the samples were assigned to is updated as the new centroid

Assignment and Update steps are repeated iteratively until the cluster assignment does not change anymore.

The similarity between objects is defined as the opposite of distance. The most widely used distance for clustering samples with continuous features is the squared Euclidean distance between two points x and y in m-dimensional space [17].

$$d(x,y)^2 = \sum_{j=1}^{m}(x_j - y_j)^2 = \|x - y\|_2^2 \quad \textbf{(2.1)}$$

The index j refers to the jth dimension (feature column) of the sample points x and y. Based on this Euclidean distance metric, we can describe the k-means algorithm as a simple optimization problem, an iterative approach for minimizing the within cluster sum of squared errors (SSE), which is sometimes also called cluster inertia:

$$SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{(i,j)}\left\|x^{(i)} - \mu^{(j)}\right\|_2^2 \quad \textbf{(2.2)}$$

In the equation above, μ is the representative point (centroid) for cluster j, and $w^{(i,j)} = 1$ if the sample $x^{(i)}$ is in cluster j; $w^{(i,j)} = 0$ otherwise [17].

To summarize, k-means is the most frequently used form of clustering due to its speed and simplicity. However, it starts with random choice of cluster centres and therefore it may yield different clustering results on different runs of the algorithm. Thus, the results may not be repeatable and lack consistency. In addition, because each iteration of *k*-means must access every point in the dataset, the algorithm can be relatively slow as the number of samples grows [26].

In conclusion of this chapter machine learning is the practice of feeding computers a huge amount of training data that the computers use to find patterns. These patterns help computers identify the correct response to various situations. There are multiple algorithms that can be used to model a data depending on the use case, most of which fall under 3 categories: supervised learning, unsupervised learning and reinforcement learning. Selecting the right algorithm is a key part of any machine learning project, and because there are dozens to choose from, understanding their strengths and weaknesses in various business applications is essential. However gathering the data is another really important stage before even choosing the ML algorithms. In the next chapter is presented the datasets of Twitter accounts used to conduct the empirical study throughout the thesis.

CHAPTER 3. PREPARATION OF THE DATA

3.1 Data acquisition

What makes Twitter data unique and more attractive for research purposes than data shared by other online social platforms is that it reflects information that users who choose to share publicly. The API platform provides broad access to public Twitter data that users have chosen to share with the world [27]. Each user can request their profiles to remain private, however, it is not the usual case. This is one of the main reasons why the thesis is based on detection of fake profiles in Twitter specifically.

The distinction of malicious users from legitimate ones using this public information is a major challenge. Therefore, the experiment in this thesis has been applied on a dataset of Twitter accounts that is collected by “the Fake project” [1]. After contacting the authors, who started the “the Fake project” in December 12, 2012, they provided their dataset for research purposes [28]. In their technical report [29], the authors have mentioned that the dataset is collected from different sources. The first source is the #elezioni2013 dataset which consist of 1481 verified accounts that belong to humans. The Fake project added more 469 human accounts from Twitter and verfied them in different way compared to the #elezioni2013 dataset. The exact acquisition is described in detail in [1]. The fake accounts are collected from three sources – they had bought 1000 fakes accounts from http://fastfollowerz.com, 1000 from http://intertwitter.com and 1000 fake accounts from http://twittertechnology.com, at a price of $19, $14 and $13 respectively. Through the Twitter APIs all the series of public information about these accounts were crawled.

All five datasets are composed of information about the user and the tweets each user had posted. Table 3.1 shows the number of accounts, tweets and relationships contained in the datasets.

Table 3.1 Statistic of the data [1]

dataset	accounts	tweets	followers	friends	total
TFP(@TheFakeProjekt)	469	563,693	258,494	241,710	500,204
E13(#elezioni2013)	1481	2,068,037	1,526,944	667,225	2,194,169
FSF(fastfollowerz)	1169	22,910	11,893	253,026	264,919
INT(intertwitter)	1337	58,925	23,173	517,485	540,658
TWT(twittertechnology)	845	114,192	28,588	729,839	758,427

In [30], the authors prove that using almost balanced training data results in the highest accuracy. Thus, for the training of the classifiers were used two of the previous mentioned datasets (Fig. 3.1), namely E13 and INT.

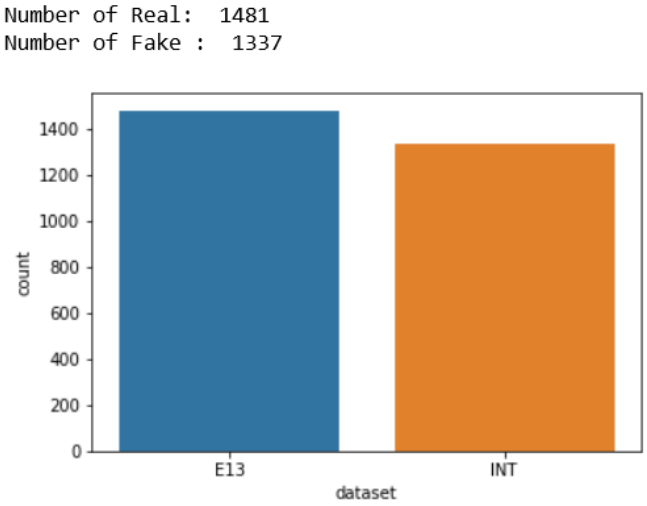

Fig. 3.1 Real and Fake datasets

All the features in each file and their explanation can be seen in Annex A: Features. The ones used for the classification and the new calculated features follow in the next subchapter.

3.2 Feature extraction

This section details the features used for training and evaluation of the performance of the machine learning algorithms.

From all 33 features available for each user, only 5 of them were chosen for classifying users as legitimate and malicious. Most of the remaining features cannot be useful since they include non-numerical data such as the URL of the profile image or the used profile background colour (Annex A: Features). Here are the public features taken directly from the Twitter API without any other calculations:

- *statuses_count*: Number of tweets every user has.
- *followers_count*: Number of followers that a user has.
- *friends_count*: Number of friends each user has. Friends are defined as the users this account is following.
- *favourites_count*: The number of tweets that a given user has marked as favourite.
- *listed_count*: The number of lists a twitter account is a member of.

In addition to this 5 features, one more feature is added which is calculated from the features available in the user's dataset:

- *FFratio:* This feature compares the number of followers to the number of friends the user has. It is calculated as: followers_count / friends_count.

From a first analysis, this feature set does not seem enough for classifying fake users. That is why some other derived features that look interesting in order to provide more clues when detecting fake users were considered. Those features are listed below and are computed from the tweets dataset (ANNEXES Annex A: Features). The average of the remaining features was calculated by dividing the sum of the values in the set by the total number of tweets.

- *URLratio*: The ratio between the tweets containing URLs to the total number of tweets.
- *average_mentions*: Average of tweets including mentioning of another user or page.
- *average_hashtags*: Average number of hashtags used in all tweets.
- *average_favorites*: Average number of times the tweets have been favourited.
- *average_retweet*: Average number of retweeted tweets the user has retweeted.
- *average_reply*: Average number of times all the tweets of every user have been replied to.

After analysing the data of the extracted new features and the public ones, the following behaviour can be concluded for the real and fake profiles. In Table 3.2 for each feature is defined below the explanation of the behaviour in brackets the minimum, the maximum and mean value for each feature in the two datasets E13 and INT.

Table 3.2 Fake profile´s detection dataset

Features(Name)	Real profile	Fake profile
statuses_count	Real profile behaviour has higher number of tweeted statuses.	Fake profiles do not tweet a lot.
	(3, 79876, 3140)	(0, 1576, 45)
followers_count	Real users have plenty of followers.	Profiles with thousands of followers are less likely to be spammers.
	(0, 408372, 691)	(0, 73, 17)
friends_count	Real profiles have most of the times a lot of people following.	Spammers do not have a large number of friends.
	(0, 12773, 403)	(0, 1998, 386)
favourites_count	Real users mark more tweets as favourites.	Fake account marks less tweets as favourites.
	(9, 44349, 439)	(0, 1402, 6.78)

listed_count	Being a member of a lot of lists is typical for real users.	Fake users are almost never members of lists
	(0, 744, 5.36)	(0, 1 0.002)
FFratio	Lower ratio values mean legitimate users	Higher ratio is more likely for fake accounts.
URLratio	Real profiles do not use often URLs when tweeting.	Bots are likely to send URLs in their tweets.
average_mentions	Real accounts uses mentioning more in their tweets	Fake accounts mention less in their tweets
	(0, 4.31, 0.68)	(0, 43.79, 0.06)
average_hashtags	Real users use often hashtags in their tweets	Fake user has less hashtags than real one.
	(0, 4.88, 0.49)	(0, 28.08, 0.2)
average_favorites	Lower value proofs real user	Higher value means malicious user
	(0, 9.9, 0.06)	(0, 33432, 32)
average_retweet	Real users tend to retweet tweets that are not so many times retweeted	Fake users retweet more famous tweets
	(0, 50847, 191)	(0, 227740, 284)
average_reply	Real profile has less replies in their tweets	Fake account has more replies in their tweets
	(0, 8.88, 0.008)	(0, 11.25, 0.06)

Further analysis was carried out by finding the correlation between all the features. The so called correlation coefficient indicates to what extent two variables are linearly related. There are several types of correlation coefficients. The most widely used one is the Pearson correlation coefficient (PCC), also known as Pearson's R [31]. This coefficient measures the degree to which a relationship between two variables can be described by a line. The original formula, developed by Karl Pearson over 120 years ago, uses raw data and the means of two variables, X and Y:

$$\rho_{X,Y} = \frac{\sum (X_i - \underline{X})(Y_i - \underline{Y})}{\sqrt{(X_i - \underline{X})^2 (Y_i - \underline{Y})^2}} \quad \textbf{(3.1)}$$

In this formulation, raw observations are centered by subtracting their means and rescaled by a measure of standard deviations [32]. The formula returns a value between -1 and 1, where:

- 1 indicates a strong positive relationship.
- -1 indicates a strong negative relationship.
- 0 indicates no relationship at all

Pearson correlation matrix was implemented with the whole new dataset containing 12 features, which was explained and discussed previous in the same section. This dataset will be used for training and testing the performance of the classifiers.

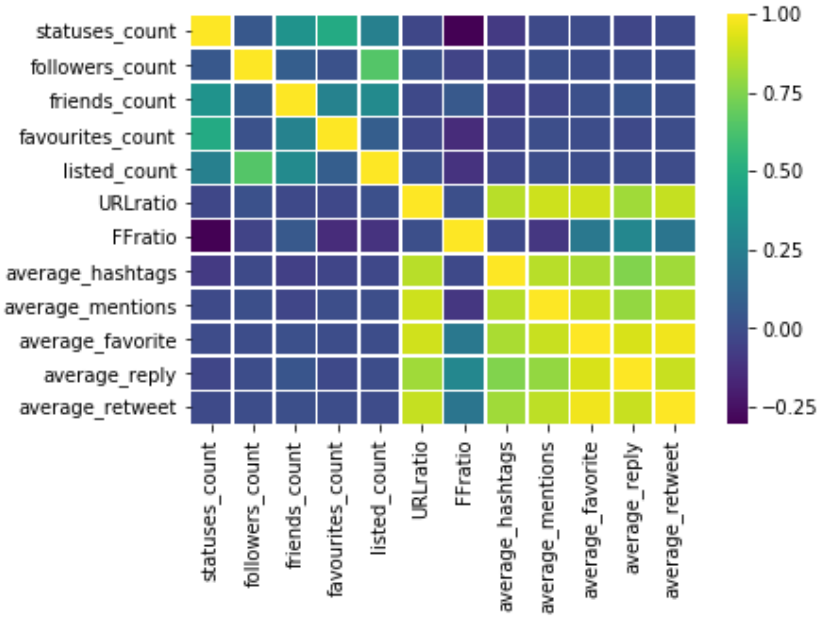

Fig. 3.2 Pearson Correlation Matrix Final Dataset

From Figure 3.3, it is clear that all the features are more or less correlated. The features calculated from the tweets are much more correlated than the ones taken directly from the user information.

3.3 Feature selection

At the beginning of the project we had four datasets with many attributes per each. In total they were 57 (Annex A: Features). Although all of the attributes were potential features, not all of them were useful for the training of the algorithms. Since the work with multidimensional data with scikit-learn library [33] is effective, it was not excluded to use simply all the attributes as feature input for the machine-learning algorithms. However, there are important reasons why reducing the number features is needed [34].

- Avoid Overfitting on Complex Models: Using many features means optimizing in a very large feature space. The greater the dimensionality, the more data is needed to be able to make statistically significant statements. Overfitting often occurs when there is too little data or the dimensions are too high.

- Reduction of training time: In addition to the size of the data set, the used algorithm and the available computational resources, the training times depend on the number of features as well.

- Improved interpretability of data: the more features you use, the harder it is to derive possible causalities. These could help you with some problems to recommend the right measurements.

Taking everything mentioned above in account, it is important to use only a limited number of features. There are some techniques and methods that can be used to estimate which features to use and which to omit. In this paper three of them are discussed and implemented - Principal Component Analysis (PCA), Recursive feature elimination (RFE) and Mutual Information (MI).

3.3.1 Principal Component Analysis

PCA is an unsupervised estimator, which means it can explore the data without reference to any known labels. Commonly it is used for dimensionality reduction, but there are much more other applications, like as a tool for visualization, for noise filtering, for feature extraction and engineering, etc. [35]

The basic idea behind PCA is to find a low-dimension set of axes that summarize the data. This algorithm helps to identify patterns in the data based on the correlation between features. In brief, PCA tries to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with less dimensions than the original one [17]. This is

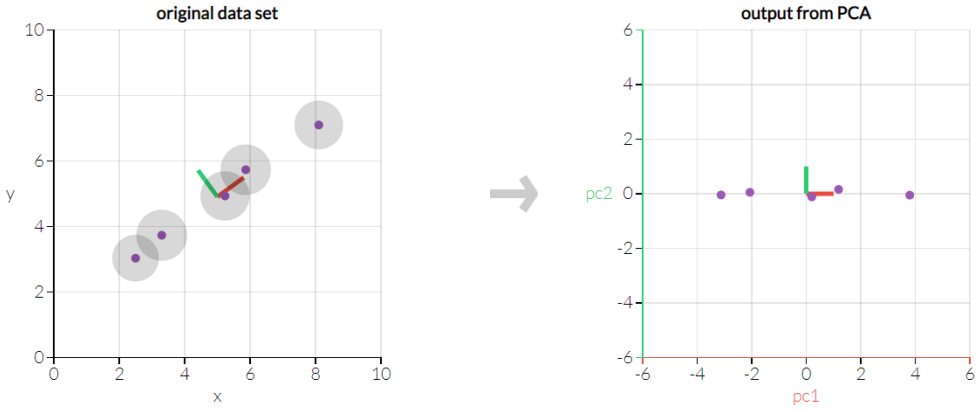

Fig. 3.3 Principal Component Analysis fundamental [63]

The orthogonal axes (principal components) of the new subspace can be interpreted as the directions of maximum variance given the constraint that the new feature axes are orthogonal to each other as illustrated in the preceding figure. x and y are the original feature axes, and pc1 and pc2 are the principal components [17].

Applying PCA to the Fake profile´s detection dataset leads to results that are the percentage of variance explained by each of the selected attributes. The sum of explained variances is equal to 1.0. The results of PCA are shown in the figure below. It is clear from the diagram that the importance is given to the public features obtained directly from the user data set (Chapter 3.2 Feature extraction). The highest variance in this case is given to the statuses_count with 0.58 and the second followers_count has 0.2. After the FFratio, which has 0.002 variance, the rest of the features have the variance of 0.

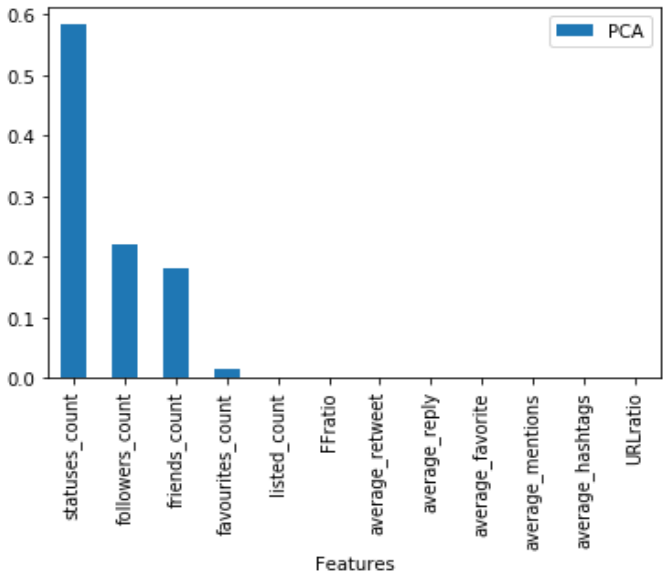

Fig. 3.1 PCA on Fake profile`s detection dataset

3.3.2 Recursive Feature Elimination

The second algorithm for feature selection is RFE. This model is based on the idea to build repeatedly a model and choose the best performing features. The process includes removing features one by one until all features in the dataset are exhausted. The ranking of the features is obtained according to when each of them was eliminated. This makes it greedy optimization for finding the best performing subset of features [36].

The model that is used for feature ranking at each iteration in the case of fake user detection is SVM. Applying the method to our feature set gives the ranking below.

Table 3.3 Recursive feature elimination

Feature	Ranking
average hashtags	1
average_favorite	2
FFratio	3
average_reply	4
average mentions	5
average_retweet	6
listed_count	7
statuses_count	8
URLratio	9
followers_count	10
friends_count	11
favourites_count	12

3.3.3 Mutual Information

Mutual information can be used as another method for feature selection [37]. It is a basic concept defined within information theory [38], which principles have been largely incorporated into machine learning.

Basically, mutual information is a measure between random variables X and Y that quantifies the amount of information obtained about one random variable, through the other random variable. It is used within the context of feature selection because it measures the relevance of a feature subset with respect to the predicting class [37]. The formula for calculating the MI is the following:

$$I(x,y) = \sum_{y\in Y}\sum_{x\in X} p(x,y).\log\left(\frac{p(x,y)}{\mathrm{p(x).p(y)}}\right) \quad \textbf{(3.2)}$$

Where p(x, y) is the joint probability function of x and y, and p(x) and p(y) are the marginal probability distribution functions of x and y respectively. If the MI is zero then the feature and the label are statistically independent and conversely, having MI of one means that the feature is strongly dependent with the output.

In the next table are presented the MI of each feature in the dataset containing real and fake user information (Chapter 3.2 Feature extraction). According to it, the average number of retweeted tweets the user has retweeted has the highest MI score followed by the average of tweets including mentioning of another user or page. On the other hand, the number of lists a twitter account is a member of has the lowest MI score.

Table 3.4 Mutual information

Mutual Information	
Feature	**Result**
average_retweet	0.678
average_mentions	0.674
FFratio	0.661
average_hashtags	0.646
URLratio	0.635
statuses_count	0.586
average_favorite	0.538
favourites_count	0.429
followers_count	0.409
friends_count	0.329
average_reply	0.314
listed_count	0.157

An advantage of this method is that it can measure even nonlinear relationship between random variables. In addition, it is invariant under transformations in the feature space such as translations, rotations, and any transformation preserving the order of the original elements of the feature vector [37].

CHAPTER 4. DETECTION OF FAKE PROFILES IN OSNs BASED ON FEATURE SET

The main focus of the thesis is to characterize and detect fake profiles in Twitter. This chapter contains the concept and the obtained results. The software of the thesis is fully implemented in Python programming language (Version 3.6.4) in Anaconda [39]. The latter is free and open-source distribution for data science and machine learning related applications. All the classification and clustering models are included in the scikit-learn [33] free software machine learning library (Version 0.19.1), which is designed to interoperate with the Python numerical and scientific libraries NumPy (Version 1.14.0) and SciPy (Version 1.0.0). Additional libraries are matplotlib (Version 2.1.2), for all the visualizations, and pandas (Version 0.22.0), for data preparation and analysis.

It is important to notice that the OS of the computer that the algorithms ran is Microsoft Windows 10 Home, the processor is Intel(R) Core(TM) i7-3517U CPU @ 1.90GHz, 2401 MHz, 2 cores and 4 logical processors. The installed physical memory (RAM) is 4.00 GB.

The rest of the chapter is structured as follows. First, the main algorithms and methods are explained. Second, the evaluation metrics and the parameters of the classifiers are defined. Finally, the evaluation is separated in 5 scenarios and the performance of the machine learning method are analyzed.

4.1 Algorithm

The dataflow procedure for detecting malicious profiles is done as follows. The raw datasets described in the previous Chapter 3 were first imported and analysed. Pre-processing is necessary, since most of the attributes are non-numerical, respectively not useful (URL of the profile, colour of the background etc.) and some of the samples include missing data.

Many machine learning algorithms do not work correctly if data is missing in a dataset. Thus, leaving missing data is not an option. From the dataset containing only directly taken features from the Twitter API, there was not any missing of the real profiles before the feature extraction, but 12 samples from the fake profiles were removed. After that feature extraction is done - from the initial set of data, new derived features are built with the intention to be informative, non-redundant and mainly not so easy to be manipulated by fake accounts. The new calculated attributes have been already discussed in chapter 3.2 Feature extraction but it is important to notice that they were considered because these are public information which can be easily extracted from a profile in Twitter. From the already discussed in chapter 3.3 Feature selection methods, some of them are applied in the scenarios as well. In the pre-processing step, the data is scaled differently according to the different scenarios. This part, together with sampling are reviewed one more time and in more detail in the first three scenarios from the evaluation part (Chapter 4.4 Evaluation) right before their use.

After the pre-processing methods, the Table 3.2 Fake profile´s detection dataset was splitted into train and test set and now the selected attributes are passed through the learning algorithms. It has to be mentioned that if there is already a new data, that can be tested, there is no need of splitting the dataset. However, in the case of detecting the fake profiles during the training phase, it is important to know which samples are real and which samples are fake users.

As already mentioned, the scikit-learn library was used for the implementation of this algorithm. In this library for each model are defined default parameters necessary for the training phase. All the defined values are listed in Annex B: Default hyperparameters. However, most of the times addition refinement of the parameters, known also like tuning, is needed for achieving higher evaluation measures. Depending on the number of the values in the tuning range, the models run again and again until finding the best combination of parameters. The exact method used in the thesis is defined in chapter 4.3.4 Scenario 4: Tuning the hyperparameters of the classifiers where tuning is first performed. In the previous three scenarios the algorithms were using the default values. The most important parameters, their explanation and the tuning range used in thesis follow in chapter 4.3 Parameters. Based on the output obtained and behaviour of these datasets the tested datasets can be predicted. The chapter 4.2 Scoring metrics follow directly after this sub-chapter.

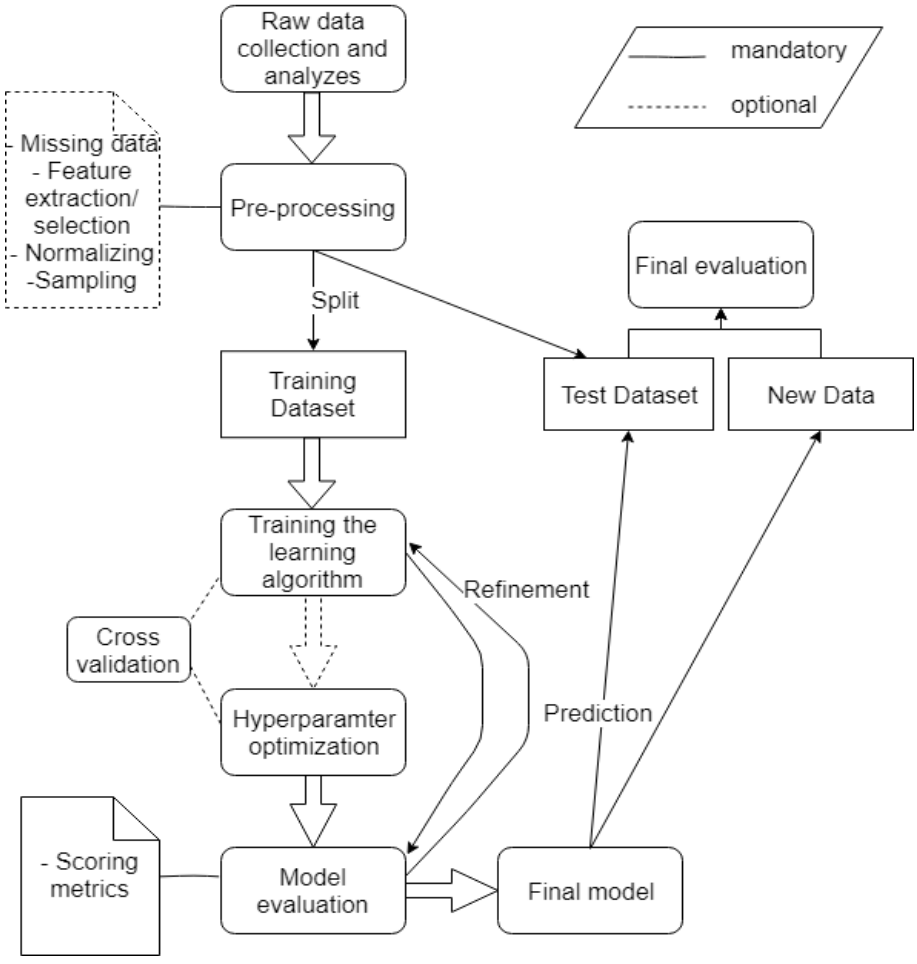

Fig. 4.1 Algorithm for detection of fake profiles

4.2 Scoring metrics

The aim of the different machine learning categories is diverse depending on the specific use case. Thus, the evaluation measures for classification tasks are different than the ones for clustering. In this subchapter are introduced all the scoring metrics used in the thesis, starting with the classification´s ones and then the clustering metrics.

Cross-validation (CV) is really useful technique to evaluate different combinations of feature selection, dimensionality reduction, and learning algorithms [64]. One of the most common is k-fold cross-validation, which is used in the thesis as well. The working principle is the following. The total amount of data is divided into k different subsets, the so-called “folds”. Typical values for k are in the range 4 to 10. In each case, one fold is selected as the test set and the remaining k-1 subsets are used for training the model. In each iteration, a model is trained with the k-1 training subsets and tested with the remaining test subset. The final performance is then averaged over the performance values of all k partitions, which gives an idea of how well the model generalizes.

There are different scoring metrics which are useful to summarize the outcomes and evaluate the performance of the classifiers, based on four standard indicators [1]:

• True Positive (TP): the number of those fake followers classified as fake.
• True Negative (TN): the number of those human followers classified as real.
• False Positive (FP): the number of those human followers classified as fake.
• False Negative (FN): the number of those fake followers classified as real.

The confusion matrix is a square matrix that reports the counts of the four indicators predicted by the classifiers, as shown in the following table. The columns are indicating the predicted number of samples and the rows are showing what the actual class of the instances is.

Table 4.1 Confusion Matrix

	Predicted class	
Actual class	fake	real
fake	TP	FN
real	FP	TN

There are three main metrics used to evaluate a classification model: accuracy, precision, recall.

Accuracy (ACC) is the most usable evaluation metric. It provide general information about how many samples are misclassified and is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the sum of correct predictions by the total number of predictions.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad \textbf{(4.1)}$$

Although ACC measures how many samples are correctly identified in both classes, it does not express if the relevant class is better recognized than the other one. Moreover, there are situations where some predictive models perform better than others, even having a lower accuracy.

Another way to interpret the accuracy is through the **misclassification error**, which is defined as: $1 - ACC$**.**

Precision (PRE) is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

$$PRE = \frac{TP}{TP + FP} \quad \textbf{(4.2)}$$

When many of the samples identified as relevant are correctly predicted, the PRE results indicated high score. However, about the samples that have not been identified as relevant PRE score is not giving any information. The latter is calculated by the recall score which shows the number of samples from the whole set of relevant samples that have been accurately recognised. Hence, lower recall score means that many relevant samples are left unidentified.

Recall (REC) is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

$$REC = \frac{TP}{TP + FN} \quad \textbf{(4.3)}$$

Sometimes is used a combination of precision and recall, the so-called, **F1-score (F1)**. It measures the test's accuracy by taking in account both the precision and the recall scores. It relationship is represented by the equation:

$$F1 = 2\,\frac{PRExREC}{PRE + REC} \quad \textbf{(4.4)}$$

With other words, the F1-score is the harmonic mean of the precision and recall, where an F1 score reaches its best value at and worst at 0 [40].

For clustering:

The scoring measures used for evaluation of the clustering algorithm are **homogeneity** and **silhouette** score [41]. A clustering result of 1.0 homogeneity means that all of its clusters contain only data points which are members of a single class. On the other hand, each cluster is represented by a silhouette score displaying which objects lie well within the cluster and which objects are marginal to the cluster. The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters.

4.3 Parameters

K-Nearest Neighbour, Support Vector Machine and Random Forest are the three supervised learning algorithms that will be used for detecting of anomalous profiles in this thesis. Each of the classifiers has its own hyperparameters. Tuning the hyperparameters is important because this directly controls the behaviour of the training algorithms and has significant impact on the performance of the model. In the following tables () are presented the parameters that are going to be tuned, its explanation, the default value in scikit-learn (for functions KNeighborsClassifier(), SupportVectorClassification(), RandomForestClassifier() and the tuning parameter range.

In Annex B: Default hyperparameters are listed all the parameters including the ones that are not tuned in the thesis.

Table 4.2 KNeighborsClassifier() tuning parameters [42]

KNeighborsClassifier()			
Parameters	**Explanation**	**Default value**	**Tuning parameter range**
n_neighbors	Number of neighbors	5	1-100
metric	Distance metric to use for the tree	‘minkowski‘	{‘minkowski‘, ‘euclidean‘, ‘chebyshev‘, ‘manhattan‘}
p	Power parameter for the Minkowski metric	2	3-5
weights	Weight function used in prediction. When uniform, all points in each neighborhood are weighted equally	‘uniform’	‘uniform’
algorithm	Algorithm used to compute the nearest neighbor. If ‘auto’, it will attempt to decide the most appropriate algorithm. Other opportunities: ‘ball_tree’, ‘kd_tree’, ‘brute’	‘auto‘	‘auto‘
leaf_size	Leaf size passed to BallTree or KDTree.	30	30

Table 4.3 SupportVectorClassification() tuning parameters [43]

SupportVectorClassification()			
Parameters	**Explanation**	**Default value**	**Tuning parameter range**
C	Penalty parameter C of the error term.	1	{0.001, 0.01, 0.1, 0.362, 1.0, 3.62, 10.0, 31.62, 100.0, 300.0}
kernel	Specifies the kernel type to be used in the algorithm.	‘rbf‘	{‘linear‘, ‘rbf‘}

gamma	Kernel coefficient for ‘rbf’.	1/n_features	{0.001, 0.01, 0.1, 0.362, 1.0, 3.62, 10.0, 31.62, 100.0, 300.0}

Table 4.4 RandomForestClassifier() tuning parameters [44]

RandomForestClassifier()			
Parameters	**Explanation**	**Default value**	**Tuning parameter range**
n_estimatos	The number of trees in the forest.	10	1-100
criterion	The function to measure the quality of a split. Gini impurity is what stays behind “gini”.	‘gini‘	‘gini‘
max_features	The number of features to consider when looking for the best split.	Square root of the total number of features	1-12
min_samples_split	The minimum number of samples required to split an internal node.	2	1-10
min_samples_leaf	The minimum number of samples required to be at a leaf node	1	1-5

Choosing appropriate hyperparameters plays an essential role in the success of every machine learning algorithm. Since it makes a huge impact on the learned model. If the learning rate is too low, the model will miss the important patterns in the data. If it is high, it may have collisions [45]. In addition, choosing good hyperparameters makes easy to manage a large set of experiments for hyperparameter tuning. The process of finding the most optimal hyperparameters in machine learning is called hyperparameter optimization and is discussed in the next subchapter 4.4 Evaluation. There the performance of the three supervised learning methods and the one unsupervised method (k-Means) is evaluated.

4.4 Evaluation

In this section all the models has been evaluated in order to determine to what extent they can detect the fake followers and how they act on predicting the target on new and future data. For this evaluation all the scoring metrics defined in chapter 4.2 Scoring metrics were used. The supervised machine learning algorithms were tested with the same data (Table 3.2), but the features were scaled differently in the first two scenarios. After that in the third scenario, the number of samples for fake and real profiles has been manipulated manually to observe their efficiency with balanced and unbalanced data. The tuning of the hyperparameters follow after that. Because future instances have unknown target values, one unsupervised method, namely k-Means, was tested in the last fifth Scenario.

4.4.1 Scenario 1: Scaling all the features equally

Standardizing the features is not only important when dealing with measurements that have different units, but it is also a general requirement for many machine learning algorithms. Since the values' range of the obtained data varies extremely, some of the machine learning algorithms will not work properly without normalization. Two of the used classifiers calculate the distance between two points to find the best prediction – kNN and SVM. When some of the features have a broad range of values compared to the rest, the distance will be governed by those particular features and, as a result, more importance is assigned to them. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance [46].

The features in this scenario were rescaled so that they have the properties of a standard normal distribution with μ=0 and σ=1, where μ is the mean and σ is the standard deviation from the mean. The formula used to get the standard scores of the samples is the following [47]:

$$z = \frac{x-\mu}{\sigma} \quad \textbf{(4, 5)}$$

In other words, for each sample of each feature the mean is removed and is scaled to the unit variance (standard deviation). However, the third algorithm used in the thesis, Random Forest, should be scale-invariant, since it belongs to the tree-based family algorithms for which feature scaling does not make huge difference in the results [47].

Before analyzing the behavior and the results from the three algorithms with and without standardizing the data, an important phenomenon has to be defined, namely, overfitting. The latter is one of the key problems in the supervised machine learning tasks.

When a learning algorithm fits perfectly the training data, so that noise and characteristics of the training data are memorized, overfitting is detected. Thus, the result of the performance drops when tested over unknown data set. Having almost 3000 samples is fundamental in this context. Small data sets are more prone to overfitting than large data sets, and even the latters are more complex, they can be easier affected by overfitting. This can lead to worsening the properties of the model, and results in untrustworthy performance when applied to novel measurements [48].

In the first scenario the behavior of the three classifiers is compared in terms of scaled and non-scaled features. In contrast to the second scenario, here all the features are scaled equally. It is worth mentioning that the algorithms were using the default parameters defined by the scikit-learn library (Annex B: Default hyperparameters). In addition the tests were obtained for three different train/test splits – 60/40, 70/30 and 80/20.

Taking all the mentioned in mind, here are the SVM accuracy results (Table ...). With scaled features the results are ranging between 0.9 and 0.92. The CV score is slightly higher than the testing score in all three splitting cases. A problem occurs without scaling them – it appears 100% overfitting. The reason is that the accuracy score for training is 1.0, but then the score drastically goes down to approximately 0.68 for the testing data.

Table 4.5 SVM-Accuracy results

SVM - Accuracy results			
Score	**Train/Test Split**	**Scaled**	**Non-scaled**
Train	60/40	0.9277	1.0
	70/30	0.9235	1.0
	80/20	0.9226	1.0
Test	60/40	0.9062	0.6742
	70/30	0.9139	0.6831
	80/20	0.9127	0.6981
CV	60/40	0.9242	0.6747
	70/30	0.9183	0.6831
	80/20	0.9158	0.6891

The next evaluation performance is for the kNN machine learning algorithm (Table 4.6). The results for the training set are slightly better than the ones for the test set for both scaled and non-scaled features. However, the difference is not so high (less than one 1%),meaning the data set was not overfitted. Compared to the training score, the testing score is getting better (from 0.9189 to 0.9254) only without scaled features for 80/20 train/test split. Using cross validation the accuracy increases with scaled features for all splits and decreases for non-scaled features.

Table 4.6 kNN-Accuracy results

kNN - Accuracy results			
Score	**Train/Test Split**	**Scaled**	**Non-scaled**
Train	60/40	0.9320	0.9247
	70/30	0.9266	0.9209
	80/20	0.9267	0.9189
Test	60/40	0.9044	0.9090
	70/30	0.9054	0.9127
	80/20	0.9072	0.9254
CV	60/40	0.9199	0.9065
	70/30	0.9141	0.9027
	80/20	0.9126	0.9062

On account of the definition of overfitting, the results using RF show light overfitting of the train set with scaled features - the test score decreases with more than 10% compared to the train score(from 0.99 to 0.88 and from 0.99 to

0.87 for 60/40 and 70/30 train/test split, respectively). Using CV validation in this case helps a lot, because the accuracy scores increases to 0.9168 and 0.9148, for 60/40 and 70/30 train/test split, respectively. Without scaling, the accuracy scores are much better. However, the test score for split 70/30 and split 80/20 are better than the CV score.

Table 4.7 RF-Accuracy results

RF - Accuracy results			
Score	**Train/Test Split**	**Scaled**	**Non-scaled**
Train	60/40	0.9908	0.9878
	70/30	0.9891	0.9895
	80/20	0.9883	0.9922
Test	60/40	0.8826	0.9153
	70/30	0.8775	0.9176
	80/20	0.9254	0.9254
CV	60/40	0.9168	0.9269
	70/30	0.9148	0.9145
	80/20	0.9135	0.9148

In consideration of the accuracy score, where the highest values were with data sliced to 60% for training and 40% for testing purposes, this split will be used from now on for the evaluation of this scenario. 40% of the data corresponds to 1099 samples, 494 of them are actual fake accounts and 605 are humans

As defined in chapter 4.2 Scoring metrics, confusion matrix is a performance measurement for machine learning classification problems, showing the effectiveness of each algorithm. The figure below contains the confusion matrices acquired after the training and prediction of samples with scaled features. For each of the four values (RP, RN, FP, and FN) the higher the number, the darker the blue color is.

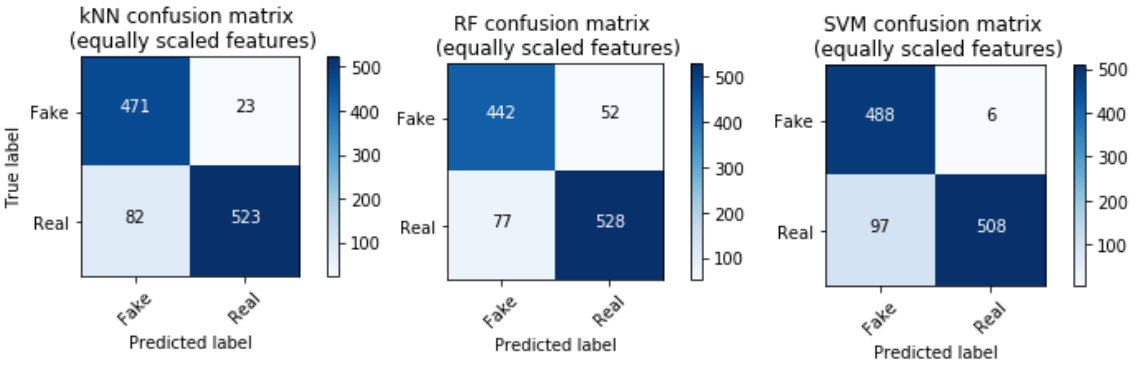

Fig. 4.2 Confusion matrices for equally scaled features

According to Figure 4.2 the CM in for KNN, there are almost 4 times more real users predicted as fake, than fake predicted as real ones. The misclassification error for this algorithm is 4.65% for the fake users (false negatives) and 13.65% for the real ones (false positives). As shown in the illustration of the RF

confusion matrix, the wrong classified fake users are slightly above 10% of the total fake accounts, but the number of false positives decreases with respect to the kNN classifier. Using SVM, the number of inaccurately predicted followers when considering the fake users is exactly 6, which means 1.2% of the total number of fake users. The difference comes with the real users. The false positive rate, real followers classified as fake, is much higher than in the previous two algorithms. The exact error is 16.03%.

As an observation, in real system, in the best case false positive values should be minimized to 0. A reason of this is that if an honest user is considered to be a fake user, Twitter will delete his/her account.

After training the data with the three machine learning algorithms, without tuning the parameters, but instead using the Annex B: Default hyperparameters, the following statements about the time execution for training and predicting can be concluded (Figure 4.3).

Random Forest shows the fastest performance in both cases, scaled and not scaled features, with an average of half second for each train/test split. When the features are scaled, kNN and SVM algorithms have almost the same execution time, between 1.0 and 1.5 seconds. The difference comes when using the original features from the Fake profile´s detection dataset without scaling them: SVM is much slower with best time execution of 4.96 seconds and worst 7.66 seconds. A reason of this is that without scaling numerical difficulties occur during the calculation. According to [49], kernel values usually depend on the inner products of feature vectors which leads to increasing the training time. On the other hand, kNN has much faster performance compared to SVM, less than a second, with non-scaled features.

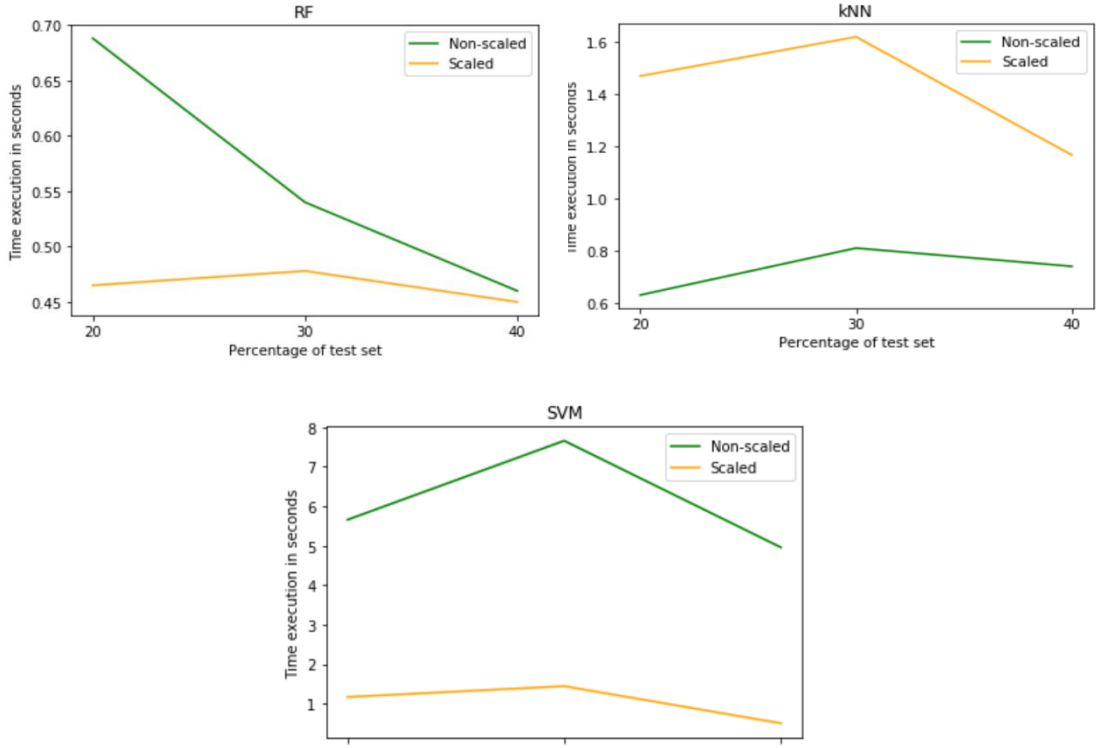

Fig. 4.3 Time execution - equally scaled features for the three different classifiers

This comparison is illustrated in Figure 4.3. The previous three line graphs, starting with the RF as the fastest algorithm, second kNN in the middle and the slowest algorithm for our data SVM, which performance is on the left hand side. It can be noticed that, except for Random Forest with non-scaled features, the time execution reaches a peak with the middle test size, namely, 30% of the whole data.

Overall, all three algorithms are fast enough when detecting the fake profiles, without even the need normalizing of the data.

Taking the overall score of the misclassification error, for all three algorithms the error is decreasing exponentially with decreasing test set. This makes sense- because less data suggests less errors. It is noticeable, that only the SVM classifier increases more than three times the error when the features are not scaled. The other two algorithms show better results without scaling the features.
Last step of the evaluation is to compare the values of the classification report. This report includes the scoring metrics precision, recall, and f1-score. Recall has to be understood as the ability of a classifier to find all relevant instances in a dataset, while precision expresses the proportion of the data points the model says was relevant actually were relevant [50] and F1-score is a combination of them. In the following table are presented the values of this report for 40% test size obtained from all three classifiers with scaled and non-scaled features.

Table 4.8 Classification report - Scenario 1

Classification report - Scenario 1			
Classifier	**Report**	**Scaled**	**Non-scaled**
SVM	Precision	0.92	0.81
	Recall	0.91	0.67
	F1-score	0.91	0.65
RF	Precision	0.88	0.93
	Recall	0.88	0.92
	F1-score	0.88	0.92
kNN	Precision	0.91	0.92
	Recall	0.90	0.91
	F1-score	0.90	0.91

As presented in the table above, scaling does not show every time better behavior of the algorithms. Excluding SVM, for which scaling is a must, RF and kNN have better precision, recall and f1-score with non-scaled features. Giving the features more importance using the original values could be an answer of those results. What exactly this means and possible increase in those scores will be examined in the next scenario. There the features are not scaled equally, but instead the values obtained in the feature selection subchapter will be taken into account.

To conclude, feature scaling can vary the results significantly while using certain algorithms and have a minimal or no effect in others. SVM is the algorithm for

which scaling is absolutely obligatory, since it achieves much better results in all scoring metrics with scaled features. RF shows exactly the opposite behavior. With non-scaled features all the metrics increases. From the performance of kNN cannot be directly concluded if scaling is necessary. In some components, the results are better with non-scaled features, but for others not. With the next scenarios, all the classifiers will be examined further, starting with assigning weights to the features.

4.3.2 Scenario 2: Scaling the features according to MI factor

An alternative approach to the standardization is the so-called **Min-Max scaling**. In this approach, the data is scaled to a fixed range. A Min-Max scaling is typically done via the following equation [47]:

$$X_{minmax} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad \textbf{(4.6)}$$

In this scenario every feature from the Fake profile´s detection dataset will be scaled to a different range, in order to assign more weights to some of them and less to the others in the case of SVM and kNN algorithm. The method used for defining the range is the one from chapter 3.3.3 Mutual Information. The results after computing the MI algorithm and the new range for the features are listed in the table below.

Table 4.9 Mutual Information Dataset

Feature	MI result	Min-Max Range
average_retweet	0.678	0-6.78
average_mentions	0.674	0-6.74
FFratio	0.661	0-6.6
average_hashtags	0.646	0-6.4
URLratio	0.635	0-6.4
statuses_count	0.586	0-5.8
average_favorite	0.538	0-5.3
favourites_count	0.429	0-4.3
followers_count	0.409	0-4.1
friends_count	0.329	0-3.3
average_reply	0.314	0-3.1
listed_count	0.157	0-1.5

The purpose of this use case is to choose which scaling for the Fake profile´s detection dataset is the most suitable for each of the three classifiers. We compare the performance of the algorithm when data is formatted in different ways. First, the result are tested with dataset that contains only non-scaled features with their original range. Second, the features in the Fake profile´s detection dataset are standardized equally as in chapter 4.4.1 Scenario 1: Scaling all the features equally. The third use case is composed of the same features, but this time they are scaled to the range in Table 4.9 Mutual Information Dataset.

For this evaluation 60% of the data is used for the training phase and 40% percentage for testing. This decision is followed by the fact that this split was showing the best results in the previous scenario.
In the table below, the accuracy results of all three classifiers for the three different scalings of the Fake profile´s detection dataset are listed. As mentioned in the first scenario, using non-scaled features for SVM leads to overfitting and scaling is preferable. However, using mutual information to assign weights to the features does not improve the accuracy scores for the SVM performance. The three scores are with half percentage worse with the Mutual Information Dataset.

Table 4.10 Accuracy results- Scenario 2

Accuracy results – Scenario 2				
Classifier	**Score**	**Non-scaled**	**Equally scaled**	**MI Scaled**
SVM	Train	1.0	0.9277	0.9211
	Test	0.6742	0.9062	0.9026
	CV	0.6747	0.9242	0.9193
RF	Train	0.9878	0.9908	0.9884
	Test	0.9153	0.8826	0.9062
	CV	0.9269	0.9168	0.9211
kNN	Train	0.9247	0.9320	0.9284
	Test	0.9090	0.9044	0.9055
	CV	0.9065	0.9199	0.9151

Although the accuracy for RF with the Min-Max scaling improves compared to the standardization, the performance of RF proves the author words in [47] and shows best results without scaling the features at all. On the other hand, kNN has better train and CV score with weights assigned to the features. Only the test score is 0.4% percentage better using non-scaled features.

A comparison of the values of the classification report obtained with the three different classifiers is presented in Table 4.11.

Table 4.11 Classification report - Scenario 2

Classification report – Scenario 2				
Classifier	**Report**	**Non-scaled**	**Scaled**	**MI scaled**
SVM	Precision	0.81	0.92	0.91
	Recall	0.67	0.91	0.90
	F1-score	0.65	0.91	0.90
RF	Precision	0.93	0.88	0.92
	Recall	0.92	0.88	0.91
	F1-score	0.92	0.88	0.91
kNN	Precision	0.92	0.91	0.91
	Recall	0.91	0.90	0.90
	F1-score	0.91	0.90	0.90

There is 1% decrease in all precision, recall and f1-score metrics using the Mutual Information Dataset. For SVM, equally scaled dataset shows the best results, while RF has better performance without any scaling. Although the accuracy scores for kNN were better with weighted features, the precision, recall and f1-score are higher with non-scaled features.

Fig. 4.4 represents how efficient kNN, RF and SVM are with the Mutual Information Dataset. As in the previous scenario, the number of tested real and fake accounts are 605 and 494, respectively.

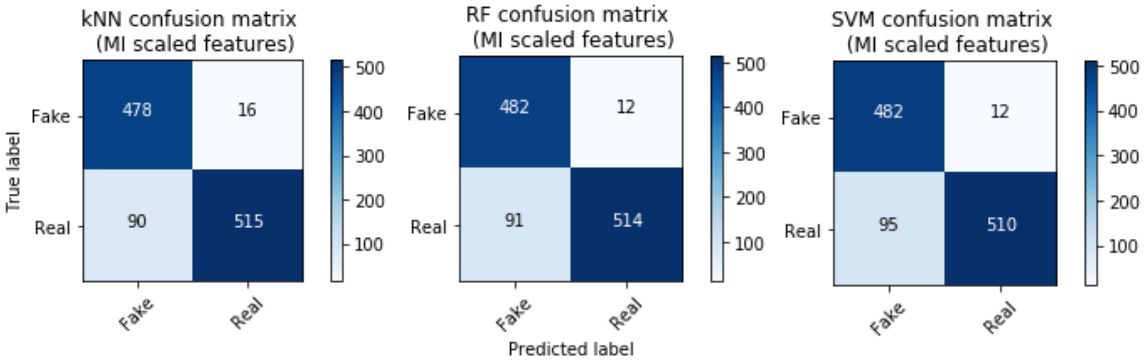

Fig. 4.4 Confusion matrices - MI scaled features

It is clear from the figure above, that there are much more real users predicted as fake, than fake predicted as real in the kNN matrix. Approximately 97% of the fake users are correctly classified, whereas the misclassification error for the real accounts is 15% using kNN for detecting the fake profiles. With RF the number of true predicted fake users' increases with four compared to kNN and the true negative value decreases with one. Using SVM the amount of correct and wrong predicted fake users remains the same like with RF. However, the percentage of accurate classification of real users decreases more and reaches 84%. In all three cases, there are much more wrong predicted real users.

There is slight difference in the results comparing this values from the confusion matrices with the obtained results from the first scenario, when the features in the dataset were normalized. kNN and RF lead to a decrease in the FN value, while SVM increases this parameter with 2 false predicted accounts. The most significant change is when using RF: 40 more fake account were classified as real (8.1% increase in the misclassification error).

In conclusion, Random Forest shows the best performance among the three classifiers without tuning any hyperparameter. It does so without scaling the features and from now on in the next scenarios when using this machine learning algorithm the features are not going to be scaled. The second best accuracy score has the SVM classifier with equally scaled features and for it exactly this dataset will be used for further examinations. The dataset composed for this scenario, namely the one with weighted features, will be used in the next scenarios only for the kNN algorithm, since with this dataset it performs better.

As next, the performance of the three algorithms will be tested with balanced and unbalanced data. What does this mean and how it can affect the performance is discussed in detail in the third scenario.

4.3.3 Scenario 3: Balanced and unbalanced data

Most machine learning classification algorithms are sensitive to unbalance in the predictor classes [51]. That is why classification accuracy can be misleading if there is an unequal number of observations in each class [52]. In the data used in the previous scenarios the number of fake users was with 14% less than the real users.
Thus, each algorithm was tested with data that has been sliced in four different partitions. In the first one the data has been reduced by randomly decreasing the number of samples with 18.97% for the real users and with 5.43% for the fake users. (1200 real samples and 1200 fake samples). This method is also known as under-sampling [51]. In the second one, the data is balanced as well, but this time the samples are much less – 200 real account and 200 fake accounts.

To observe the performance with balanced and unbalanced data the third use case contains approximately 17% fake users and the rest to 100% is filled with data of the real users. The fourth, final use case includes exactly the opposite distribution, namely 200 real users and 1200 fake users.

Table 4.12 Accuracy results for balanced and unbalanced data

Accuracy results for balanced and unbalanced data					
Classifier	**Train/Test splits**	**real = fake (1200 - 1200)**	**real = fake (200 - 200)**	**real > fake**	**real < fake**
SVM	Train	0.9993	1.0	0.9095	0.8691
	Test	0.9958	0.975	0.9846	0.8571
	CV	0.9958	0.9833	0.8904	0.8678
RF	Train	1.0	1.0	0.9952	0.9902
	Test	0.9989	0.9937	0.9107	0.9162
	CV	0.9986	0.9916	0.9	0.9217
kNN	Train	0.9937	0.9917	0.8631	0.9131
	Test	0.9937	0.9937	0.8411	0.8696
	CV	0.9901	0.9916	0.8321	0.8726

Table 4.12 outlines the best accuracy results with each of the four distributions for the three machine learning algorithms which were using the Annex B: Default hyperparameters. From first sight the results are significantly good, especially when the data is 100% percentage balanced. RF and SVM have even 100% accuracy for the train sets. The test and CV scores are excellent as well. All three classifiers have an increase and reach more than 99% percentage accurate predictions. The amount of samples seems to be not so important since with 83% percentage less data all three scores are almost the same and exceeding 97% correctly classified samples.

Furthermore, the accuracy obtained for the performance of the three algorithms with unbalanced data seems to decrease substantially, although it seems to be pretty high. The worse result has kNN on the CV score with 0.8321 and the best one has RF on CV score with 0.9217. As already mentioned the accuracy results can be confused with unbalanced data. Moreover the dataset will bias the prediction model towards the more common class! That is why it is important to observe the results of the other metrics.

In the following table are presented the results of the precision and recall scores obtained with the four different sample´s distributions. The results are specified for real and fake users in order to observe easier how the unbalanced data contributes the performance of the classifiers.

Table 4.13 Precision/Recall scores for balanced and unbalanced dataset

Precision/Recall for balanced and unbalanced data					
Classifier	**Classification report**	**1200 real 1200 fake**	**200 real 200 fake**	**1200 real 200 fake**	**200 real 1200 fake**
SVM	Fake	0.99/1.0	0.95/1.0	1.0/0.26	0.86/1.0
	Real	1.0/0.99	1.0/0.96	0.89/1.00	0.0/0.0
	Total	1.0/1.0	0.98/0.97	0.91/0.89	0.73/0.86
RF	Fake	1.0/1.0	0.99/1.0	0.77/0.54	0.92/0.97
	Real	1.0/1.0	1.0/0.99	0.93/0.97	0.71/0.5
	Total	1.0/1.0	0.99/0.99	0.9/.91	0.89/0.9
kNN	Fake	1.0/0.99	1.0/0.99	0.29/0.07	0.89/0.97
	Real	0.99/1.0	0.99/1.0	0.86/0.97	0.59/0.29
	Total	0.99/0.99	0.99/0.99	0.78/0.84	0.85/0.87

Again with balanced data the results are outstanding. The precision and recall scores are 0.99 and 1.0 for all three algorithms. The differences are clearly observed when the amount of real users and fake users is highly diverse. The total scores can be misguided, since they are noticeably high. Both of the precision and recall sores are calculated through the four values obtained in the confusion matrix and for better understanding of the high fluctuations they are presented in Table 4.14.

For better visualization and analysis of the results, the four values that the confusion matrix contains (true positive, true negative, false positive and false negative) are presented in a table.

The actual number of the tested samples corresponding to fake and real accounts, when having equally 1200 instances each, is 478 fake and 482 real. The reason is that the train/test split is 60/40. With less data, but again exactly balanced, the number of fake users that has to be predicted is 70, compared to the 90 real users. When dealing with the unbalanced data, the distribution real/fake users is exactly 40% percentage of the number of samples - 480 and 80, respectively.

As already seen in the accuracy results, with the same amount of fake and real users, there is almost no misclassified samples. Using SVM only 3 real users have been predicted as fake when the number of instances is 1200 each. RF has only one wrong predicted account and the highest number (FN=6) of misclassification error has the kNN algorithm - 1.25% percentage. Less but again balanced data leads to 4.44% percentage of error with SVM. The other two classifiers have only 1 wrong predicted value – for RF the false positive value is 1 and for kNN the false negative value.

The table data clearly shows the difference in the results when having unbalanced data. If real users are more than fake users, SVM is not making any error classifying the real ones, but the right fake predicted accounts are only 26.25%. The bigger problem comes when the real number of samples are less than the fake ones. In this case, using SVM all the users are classified as fake users. RF shows the best results with unbalanced data predicting correct more than 50% of all samples even the minority class. The third algorithm, kNN, has better performance when the fake accounts are more than the real accounts.

Table 4.14 Confusion Matrix Values for balanced unbalanced data

Confusion Matrix Values for balanced and unbalanced data						
Number of samples		**Classifier**	**TP**	**TN**	**FP**	**FN**
Real	**Fake**					
1200	1200	SVM	477	479	**3**	**1**
		RF	478	481	1	0
		kNN	472	482	0	6
200	200	SVM	70	86	4	0
		RF	70	89	1	0
		kNN	69	90	0	1
1200	200	SVM	21	480	0	59
		RF	42	467	13	37
		kNN	6	465	15	74
200	1200	SVM	480	0	80	0
		RF	464	40	40	16
		kNN	464	23	57	16

To summarize, the distribution between the classes in a dataset can lead to high changes in the performance of the machine learning algorithms. Although it is preferable to have equal number of real and fake users in the data set, in a real scenario this is almost impossible to be achieved. The results with unbalanced dataset were not promising, but in this situations there is another method for increasing the accurate prediction, namely tuning the hyperparameters of each classifier.

4.3.4 Scenario 4: Tuning the hyperparameters of the classifiers

Defining the model architecture of any machine learning algorithm is one of most important thing for having good results. Often, the optimal model architecture for a given model is not known immediately, and thus an exploration of the possibilities is necessary. The optimal way is to let the

machine to perform this exploration and select the best model architecture automatically. Hyperparameters are called the parameters which define the model and thus the process of searching for the ideal model architecture is known as hyperparameter tuning.

For the exploration in this thesis is used the so called grid search method [53]. It is one of the simplest strategies, in which all possible combinations of given discrete parameter spaces are evaluated.

Each classifier model has different hyperparameters and in the next three subsections they will be explained in detail. The Table 3.2 Fake profile´s detection dataset is used for all the algorithms. However, as a result from the previous scenarios different scaling of the features was computed. For SVM all the features were standardized equally. kNN uses the Table 4.9 Mutual Information Dataset and for RF no scaling was computed. In addition, the changes in the performance depending on the values is shown. Then, a comparison will be made and the best model performance will be chosen.

4.3.4.1 Support Vector Machine

The aim of standard SVM is to find a hyperplane that separates all positive from negative examples. However, in case of mislabeled or extremely different samples, this leads to not fitting well the models. Using a soft margin constant allows those samples to be "ignored" or placed on the wrong side of the margin. This innovation can lead to a better overall fit. So, the first important hyperparameter is exactly this soft margin constant C, which controls the influence of each individual support vector [54].

Depending on the type of the kernel function, e.g. the width of a Gaussian kernel (gamma) and the degree of a polynomial kernel, are defined the other important hyperparameters. The explanation of its impact on the decision boundary follows.

To begin with, the soft-margin constant and its effect are illustrated in the Figure below. The C value is common to all SVM kernels and trades off misclassification of training examples against simplicity of the decision surface. A low C makes the margin larger, while a high C aims at classifying all training examples correctly with decreasing the margin [55].

The next hyperparameter that has a significant effect on the decision boundary is the kernel. The function of kernel is to take data as input and transform it into the required form, with other words - it converts not separable problem to separable problem. Different SVM algorithms use different types of kernel functions and in [56] all the types their potential uses are explained.

For the detection of fake profiles only two kernel functions are used– linear and Gaussian radial basis function (RBF). The latter is one of the most widely used kernels:

$$k\left(x^{(i)}, x^{(y)}\right) = \exp\left(-\gamma \left\| x^{(i)} - x^{(y)} \right\|^2\right) \qquad \textbf{(4.7)}$$

The free parameter that can be optimized is γ. The term kernel can be interpreted as a similarity function between a pair of samples. The minus sign inverts the distance measure into a similarity score and, due to the exponential term, the resulting similarity score will fall into a range between 1 (for exactly similar samples) and 0 (for very dissimilar samples) [17].
In the figure below is illustrated the impact on the decision boundary of the gamma parameter for a fixed value of the soft-margin constant. The four diagrams reveal that for small values of γ the decision boundary is almost linear. With increasing γ, the flexibility of decision boundary increases as well. Hence, higher values of γ can lead to overfitting [55].

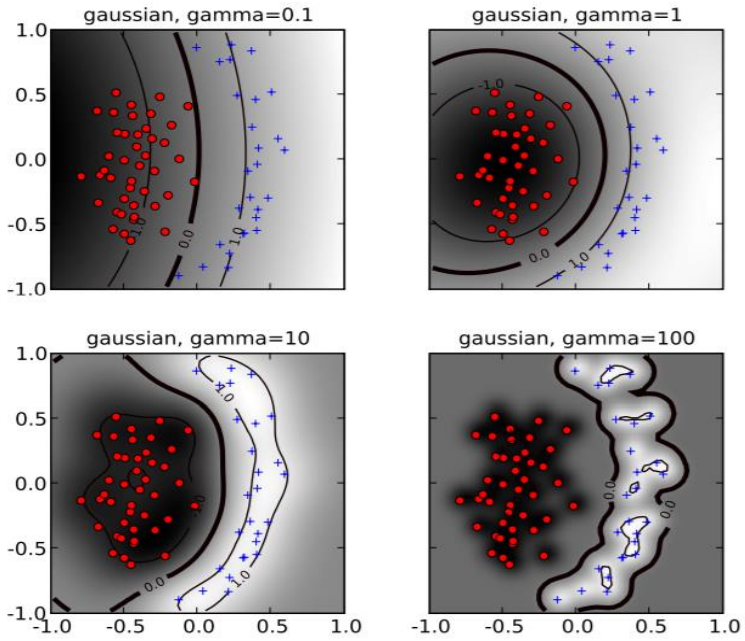

Fig. 4.5 RBF gamma parameter [55]

With the help from the grid search method both kernels (linear and RBF kernel) were tested on the dataset with real and fake users. The range of values for the parameters C and gamma is 10 values between 0.001 and 300 (Chapter 4.3 Parameters). The time for computing all the possible combinations and finding the best one was 646 seconds. The combination of hyperparameters, for which the CV accuracy score is the highest, is with RBF kernel, soft margin constant equal to C=31.62 and 0.1 as value for the gamma parameter. The CV score they reach is 0.9277.

Before analyzing the results for the linear kernel, Figure 4.6 provides the changes of the accuracy depending on the gamma and the C parameter. Only four values of the C parameter are examined further – 0.01, 0.1, 10.0 and 100.0.

As it is presented in the two line graphs below, the higher CV score is achieved for small values of gamma. Overall, with increasing gamma the accuracy decreases for all four C values. For smaller values of the soft margin constant the drop is more rapid, while for larger values it is gradual. The worst accuracy, 0.5297, was achieved for C equal to 10 and 100, when gamma is 300.

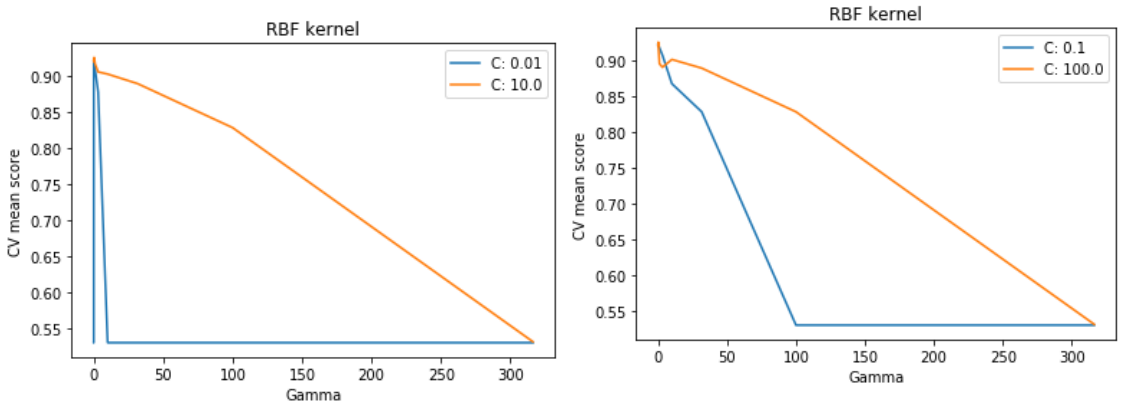

Fig. 4.6 Relation - soft margin constant vs. gamma parameter

The dataset with information about real and fake users seems to be linearly separable. The SVM performance with linear kernel is reasonably satisfying. Figure 4.7 compares the CV accuracy only for the small values of C, since with higher values than 0.1, the accuracy score fell down and remain steady. The highest score, 0.9229, is achieved with soft margin constant equal to 0.01. However, this score is not better than the one with the RBF kernel.

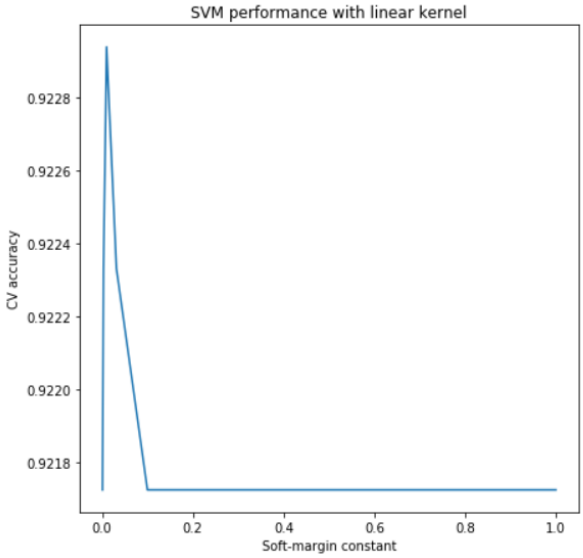

Fig. 4.7 SVM performance with linear kernel

To sum up, both linear and RBF kernel SVM have their own advantages and disadvantages. RBF kernel shows better results. On the other hand, linear SVM is a parametric model, while RBF kernel SVM is not. Thus, the complexity of the latter grows with the size of the training set. Furthermore, RBF kernel has more hyperparameters, so model selection takes more time and is more expensive [57].

4.3.4.2 K-Nearest-Neighbor

For a standard kNN implementation, there are two primary hyperparameters that are important to be tuned - the number of neighbors and the distance metric function. Both of these values can dramatically affect the accuracy of the kNN classifier.

In order to find which is the best k that corresponds to the lowest test error rate, first a list with possible k values is created. The list consist only of odd numbers to prevent tie situations. After performing a 10-fold cross validation on the dataset using the generated list, the results were analyzed and the misclassification error was calculated for all the iterations. The lowest one is the optimal k. The used dataset for kNN is the one generated for the second scenario with weighted features. Four different distance metrics were used for the testing of the kNN performance – euclidean, manhattan, chebyshev and minkowski distance. The four following line graphs show the misclassification error for a range of 100 k values for the four distance metrics. It is clear from the graphs that the misclassification error is higher for lower k values and with higher than 10 neighbors the result fluctuates.

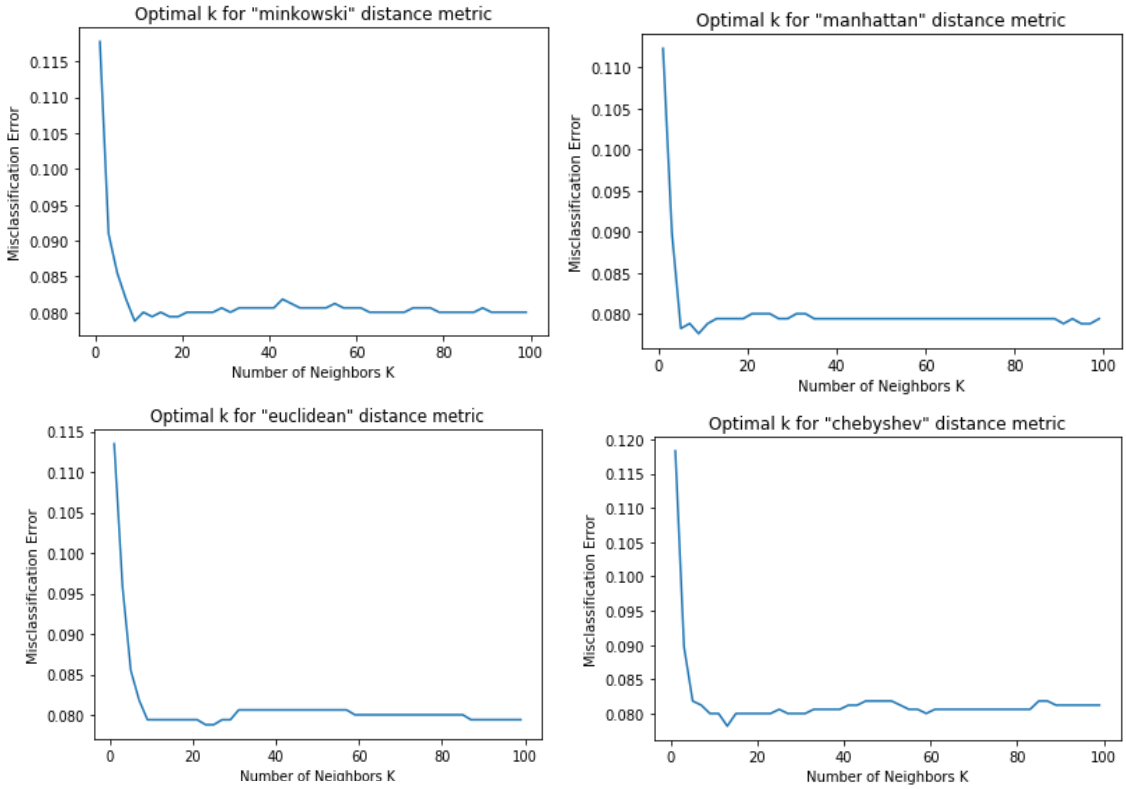

Fig. 4.8 Relation between number of neighbours and distance metric

The optimal k for each of these distances and the train, test and CV scores are represented in the table below.

Table 4.15 kNN tuning evaluation

number of neighbors vs distance metric				
Distance metric	**Optimal k**	**Train**	**Test**	**CV**
euclidean	23	0.9217	0.9054	0.9199
manhattan	9	0.9247	0.9072	0.9217
chebyshev	13	0.9217	0.9054	0.9199
minkowski	9	0.9229	0.9063	0.9193

Clearly seen from the scores, the best combination is manhattan distance metric with 9 neighbors. For other two of the combinations (distance metric, optimal k) the obtained results for all three scores are absolutely the same. These are euclidean metric with 23 neighbors and chebyshev metric with 13 neighbors. The last combination (minkowski metric with 9 neighbors) has the second best scores

Overall, it is important to examine for each of the distance metrics the optimal k since the results can vary significantly.

4.3.4.3 Random Forest

Random Forest is a tree-based algorithm, popularly used in all kinds of data science problems and is an ensemble of decision trees. Decision tree is a type of supervised learning algorithm that splits observations into the classes based on the best splitter at each step. However, decision trees can suffer from high variance which makes their results fragile to the specific training data used.

One way to reduce this variance, is to tune its hyperparameters, forcing the trees to be different leading to more accurate and stable predictive performance. [58] The default function to measure the quality of split used for all the examinations is the Gini impurity.

The first hyperparameter that can be tuned to improve the predictive power of the model is the number of trees to be build. For this evaluation all other parameters of RF were fixed with the default values [59]. The following line graph gives information about the changes of the CV score according to the number of trees or estimators.

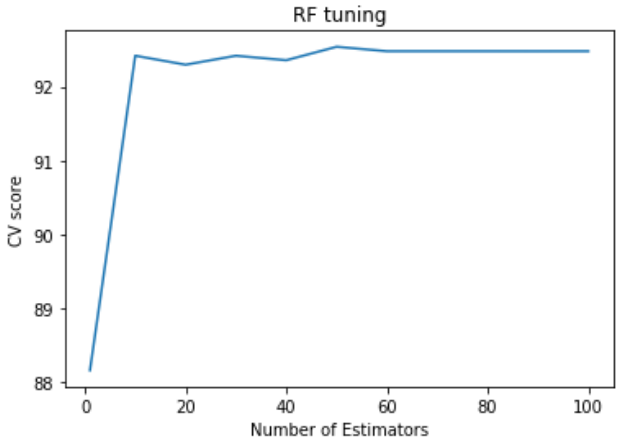

Fig. 4.9 RF tuning - Number of estimators

For up to 10 numbers of trees the CV accuracy has a dramatic growth of more than 4%. Then, for the next possible values up to 60 it fluctuates and after that it stabilizes until reaching 100 trees. However, the highest score RF achieves with 50 numbers of trees.

The next parameter is the number of features to consider when looking for the best split. In addition to the other default parameters, this time is added the number of estimators to be 50, with which RF has the highest CV score. Having

in mind that the number of features in the dataset is 12, in this evaluation were tested all possible numbers from 1 to 12 and the result is illustrated in Figure 4.10.

With only one feature per split the CV score is considerably high. Then it starts to fell gradually until it reaches a bottom of 92.3% with 4 features. After a quick increase RF reaches the peak of more than 92.6% with 5 features per split. Again a dramatic decrease follows and the worse accuracy is achieved with 9 features. Afterwards, the CV fluctuates while it reaches a moderate result for 12 features.

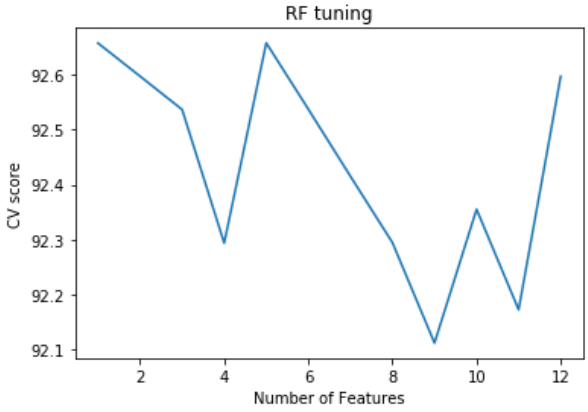

Fig. 4.10 RF tuning - Number of features

The minimum number of samples required to split an internal node is another hyperparameter which can make predictions of the model better. Again, as in the previous example, to the other parameters is added the obtained result for the maximum number of features, namely 5. This parameter can vary between considering at least one sample at each node to considering all of the samples at each node. When this parameter increases, each tree in the forest becomes more constrained as it has to consider more samples at each node. The range in minimum sample´s split parameter for which the performance of RF was tested is from 10% from all the samples to 100% of the dataset.

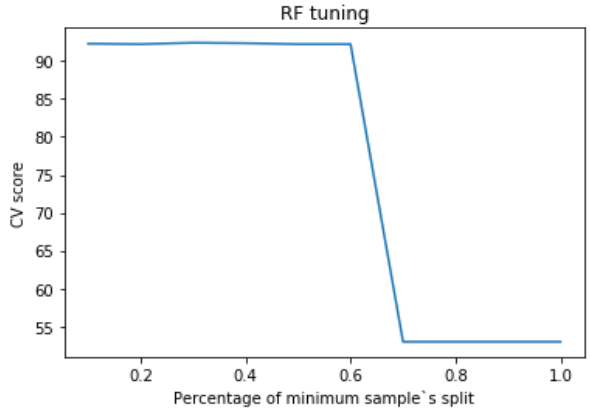

Fig. 4.11 RF tuning - Minimum sample´s split

The last parameter important for defining the model architecture of RF is the minimum number of samples required to be at a leaf node. When this parameter increases, it leads to a decrease in the variance and an increase of the bias. So, this parameter controls the level of regularization when growing

the trees. It is worth noticing that increasing this value can cause underfitting [60].

The next line graph illustrates the CV score results with different sample's leaf ranging from 10% of the samples to 50% of the samples. The best accuracy is achieved with 20% of the samples to be at a leaf node.

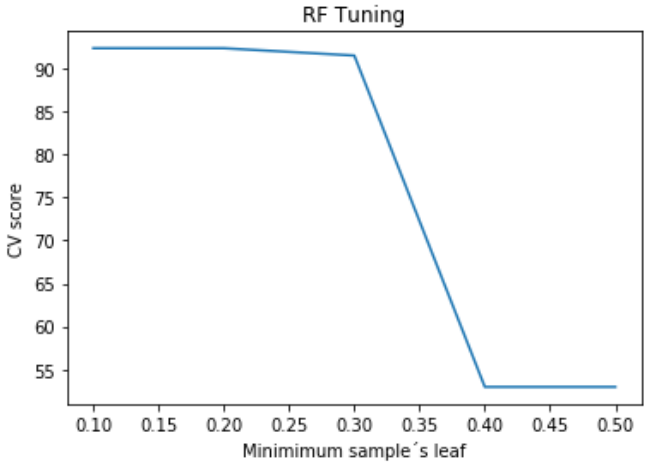

Fig. 4.12 RF tuning - minimum sample´s leaf

To conclude, the best hyperparameters are 50 number of estimators, with 5 number of features, 60% of the samples required to split an internal node and 20% required to be at a leaf node.

4.3.4.3 Comparison

Here is presented the difference in the accuracy when the hyperparameters of the three classifiers were tuned with the best possible combination obtained in the previous sections and the combination of the Annex B: Default hyperparameters. With tuning is observed slight increase in CV score of SVM and kNN, whereas for RF the score remains the same. The test score is decreasing with 1% for SVM, but is increasing for the other classifiers.

Table 4.16 Accuracy results comparison

Accuracy results comparison			
Classifier	**Score**	**Tuned parameters**	**Non-tuned parameters**
SVM	Train	0.9382	0.9277
	Test	0.89	0.9062
	CV	0.9278	0.9241
RF	Train	0.9229	0.9902
	Test	0.9044	0.8817
	CV	0.9212	0.9215
kNN	Train	0.9247	0.9320
	Test	0.9072	0.9044
	CV	0.9217	0.9199

4.3.5 Scenario 5: Applying unsupervised learning algorithm

The aim of the unsupervised learning approach is to recognize unknown patterns from the data and to derive rules from them. The differentiation between the real and fake user activity is difficult since every fake account can manipulate easy most of the real user activity. In such situation, when the data is not labelled, unsupervised learning is very often used. The advantages of this method are the partially fully automated creation of models. These can produce a very good prognosis about new data or even create new content. The model learns with each new record and at the same time refines its calculations and classifications. Manual intervention is no longer necessary [61].

Applying clustering algorithm to the real and fake user's dataset should create two groups of data points – one for the real users and one for the fake users. The points in different clusters are dissimilar while points within a cluster are similar. K-Means is the algorithm used in this scenario. However before applying it, another unsupervised learning method was implemented, namely 3.3.1 Principal Component Analysis. The reason is that PCA serves as a dimensionality reduction method on the features of the original dataset by projecting these features onto a lower dimension. Therefore, from the original dataset which contains 12 columns (i.e. features) it reduces them down to 2 columns. This is illustrated in the left diagram of Fig 4.13.

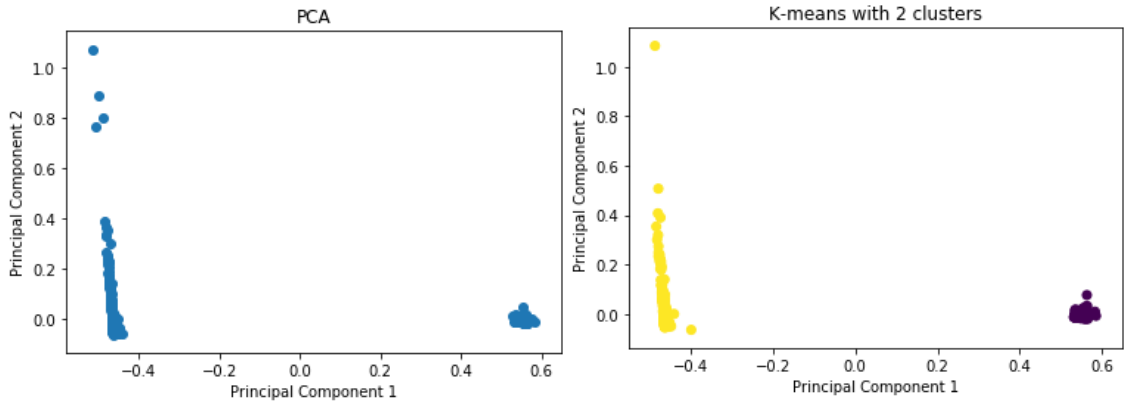

Fig. 4.13 Unsupervised learning evaluation

After the dimensionality has been reduced, the separation of the clusters is directly visible. Proving the expectations, after applying the k-means algorithm the two groups were separated as presented in the right diagram of Figure 4.13. In this case, the right labels are available and it is possible to calculate the accuracy of the k-means performance – 100%. All the fake users were classified as fake ones and analogically the real accounts were predicted correctly. Two other scoring metrics are important for the evaluation of unsupervised learning and they are silhouette score and homogeneity score and the obtained results for them are 0.91 and 1.0 respectively.

To sum up, using clustering algorithm for detecting fake profiles can be really beneficial and useful. Normally, there is no general l solution to find the optimal number of clusters for any given data set when using k-Means. However, in this case it is clear, that the clusters are two and this makes it ideal use case for clustering analysis.

CHAPTER 5. CASE STUDY: TRUMP'S FOLLOWERS

Just as with all other aspects of modern life, the internet is quickly becoming interwoven into political campaigns, creating a new form of “smart” politics. This is the reason why in the final chapter of this master thesis part of the followers of the forty-fifth president of the United States, Donald Trump, were used as test set for predicting his real and fake followers.

President Donald Trump entered the Time magazine [51] list of people who have the most influence on the Internet. The list is published on the site of the publication and features people who have a "global impact on social media" and also have the ability to set the tone and direction of world news. The publication defines Trump as "the world leader with the highest number of followers on Twitter, which is an extremely effective tool for presenting his messages. In Twitter he is known as @realDonaldTrump and has very strong presence. He has the account since March 2009 and made it to having already 54,892,772 Twitter followers [49]. This number is ranking him 17th among all Twitter users. @realDonaldTrump tweets very frequently, with an average of 11.84 tweets per day in the past 30 days (from the 5th of September 2018 to the 5th of October 2018) and total of 39,172 since @realDonaldTrump joined Twitter. It seems like Donald J. Trump is really being listened to on Twitter, with an audience attentiveness score of 72%, which stems from being tracked on 94,457 Twitter lists [50].

The machine learning models are going to be learned with the data used in the previous chapters and only tested with the new crawled data. This data of the followers who are going to be examined was collected through the Twitter API. In total the accounts are 805 and all of them are containing data only of their user feed and not any details about the tweets. However, when PCA was performed (Chapter 3.3.1 Principal Component Analysis) on the whole dataset exactly the following features were ranked as the most useful ones. The table below gives information about the minimum, mean and the maximum value for each of the five features from the Trump´s dataset.

Table 5.1 Trump´s dataset

	followers_count	**friends_count**	**listed_count**	**favourites_count**	**statuses_count**
min	0.000000	0.000000	0.000000	0.000000	0.000000
mean	2.284472	82.592547	0.007453	49.658385	8.101863
max	29.000000	757.000000	1.000000	28928.000000	709.000000

Direct from the first sight is clear that this dataset is much different than the Table 3.2 Fake profile´s detection dataset. As an example, the mean number of statuses for real users is 3140 and for fake users is 45. In this case study, this value is only 8. The difference is in all the other features. Therefore, further analysis of the data was needed.

What was noticed is that all the accounts were 2 months old, when the data about them was collected. On the other hand, the training data consist of profiles that are at least 1 year old and even some of them had been created more than 5 years in advance.

This is the main reason of the obtained results after the prediction of Trump´s followers. These are shown in the following table. The dataset was scaled to the range between -1 and 1 and with it the three models, SVM, RF and kNN, are trained once with their Annex B: Default hyperparameters and a second time with the best combination of parameters obtained from the evaluation part in the previous chapter for each model.

Table 5.2 Rate of fake Trump's followers

Rate of fake Trump's followers		
Classifier	**Scaled features**	
	Non-tuned	**Tuned**
SVM	32.9	39.5
RF	46.9	46.3
kNN	53.6	38.7

The models predict at least 32% of these 805 profiles to be fake. SVM and kNN have significant difference of the predictions with tuned and non-tuned parameters. RF, on the other hand, shows balance in the meaning that in both cases it classifies 46% fake profiles.

After that was analysed which accounts exactly were predicted as fake and it was clear that the training and the testing set are not following the same pattern. For example, accounts with more followers and friends are classified as fake ones and profiles without even one follower are detected as real, which does not make sense in real scenario. However, this comes from the fact that in the training data the real and fake accounts have an average of 690 and 17 followers, respectively. In the Trump´s dataset the mean value for this feature is 2, which is much more close to 17 than to 690 followers.

Another important thing that it is good to refer is that as already mentioned in chapter 1.2 Fake profile, this year Twitter deleted 70 million fake profiles in May and June [3]. The profiles that were tested in this chapter were created shortly after this period. Most likely this is a reason why so many new profiles were created.

In this regards, it can be concluded that to the real accounts should be given time to develop and expand their profiles. Otherwise, they will be detected as fake and removed from Twitter, which is not acceptable.

All the assumed conclusions and possible future work are presented in the last final chapter.

CONCLUSIONS

Fake followers are those Twitter accounts specifically created to inflate the number of followers of a target account. Fake followers are not only dangerous for the social platform, since they increase popularity and influence in Twitter — hence impacting on economy, politics, and society. Therefore, the focus of this master thesis was to find efficient techniques for fake Twitter followers' detection and to evaluate their performance.

Machine learning has become an integrative part of the modern scientific methodology, offering automated procedures for the prediction of a phenomenon based on past observations, finding underlying patterns in data and providing insights about the problem. For the thesis were analysed three classification algorithms – SVM, RF and kNN – and one clustering algorithm, k-Means.

The developed machine learning based approaches are based on user profile activities and their communication with other users. These activities were characterized through a feature set of 12 features covering different specifications of their tweets, such as their amount, likes, usage of URL and hashtags, mentioning other users and activities based on the friends and followers of the account that is under investigation. The dataset with the Twitter profiles was containing 1950 real users and 3351 fake users in total. The total amount of available tweets was 2 827 757. However, for having more or less balance between the classes for the training were used 1481 real profiles and 1337 fake profiles.

Using three feature selection methods – PCA, MI, RFE – the feature based dataset was examined further and the three different approaches ranked as most important different features. The total number of tweets, the average number of retweet messages and the average hashtags used per tweet were the best ranked features for PCA, MI and RFE, respectively.

Training the three classifiers with this feature based dataset led to efficient separating the real profiles from the fake profiles with achieving accuracy of 92%. This results were obtained with tuning their hyperparameters which caused an increase in the accuracy of approximately 0.3% for each of them. The performance of the classification algorithms were examined further, trying to find which scaling of the feature is the most appropriate. The results shows that Random Forest has the best results with using the original range of the features. The SVM classifier perform the most accurate when all the features are normalized equally in the range between -1 and 1. The third classifier, kNN shows its best results when to the features are assign weights, according to the MI factor.

Furthermore, the thesis proves that the different balance between the classes in a dataset results to fluctuations in the performance of the machine learning methods. Although it is better to have equal number of real and fake users in the dataset, in real life this is really difficult achievable. The reason is that the real profiles are much more than the fake profiles.

The clustering algorithm that was tested is k-Means, which was able to detect all the fake accounts. This result is promising that unsupervised learning is the right method for detection techniques over user behaviour to distinguish potentially bad behaviour from normal behaviour.

The effective employment of fake profiles´ detection approaches, will enable Twitter and other similar OSNs to maintain a platform that is populated with real users and, thus, be a beneficial tool for accurate data gathering.

From the analysis and conclusions derived from the work on the thesis, some aspects are worth to be explored further in order to continue the analysis on the possibilities for detecting malicious users not just in Twitter but also in other OSNs with similar characteristics.

Machine learning technology typically improves efficiency and accuracy over time thanks to the ever-increasing amounts of data that are processed. This gives the algorithm more “experience,” which can, in turn, be used to make better decisions or predictions. In this regard, collecting more extent dataset, analyzing the differences and how to deal with them are the next main steps. If the trained data keeps increasing then the accuracy in classifying the data sets also increases.

Further step is to understand the behavior of fake users whose creation aim is different. More precisely to explore the possibility of distinguishing among the different kind of malicious profiles used for cybercriminals such as spammers, hacked profiles, duplicated profiles etc. With the help of unsupervised learning is possible to discover hidden patterns in data and hence to define different manners.

Sustainability and environmental awareness: In today's digital world, popularity in social networks plays an essential role in consumer behavior, public and eco-supporters, and even in investor decisions and political motions. Detecting and removing false profiles will limit the potential for malicious influence on public attitudes, which will lead to a near-realistic picture of social, economic, social and political behavior.

Ethical implications technology: The data used in the thesis is analyzed only for research purposes and the used features are only numerical. Online social networks have suffered from a wide range of threats to users’ security and privacy for years. One key threat is fake profiles which are often the root of online social network evils. These fake users can collect dozens of personal details about real users and their friends. The security of the personal data and the correctness of the information received and shared is a major factor in the existence and development of a social network. Therefore, removing fake profiles is a necessary improvable measure order to maintain consumer faith.
Detecting them does not threaten the freedom of speech but guarantees access to not manipulated information.

REFERENCES

[1] Cresci S, Pietro RD, Petrocchi M, Spognardi A, Tesconi M. “Fame for sale: Efficient detection of fake Twitter followers.” Decision Support Systems. (2015) Pp.(1, 2, 13, 14)

[2] Madhav, A, and Reddy, VSN, “Automated detection of fake profiles using simple framework: SVM”, International Journal of Advance Computing Technique and Applications (IJACTA), None, 2016, Pp. 1

[3] “Twitter is sweeping out fake accounts like never before, putting user growth at risk,” *The Washington Post*, 06-Jul-2018. [Online]. Available:https://wapo.st/2IK4V4h [Accessed Oct. 9, 2018]

[4] “About,” *Twitter*. [Online]. Available: https://about.twitter.com/en_us.html [Accessed Oct. 9, 2018]

[5] “Internet Live Stats - Internet Usage & Social Media Statistics,” *Google Search Statistics - Internet Live Stats*. [Online]. Available: http://www.internetlivestats.com/ [Accessed Oct. 9, 2018]

[6] A. El Azab, A. Idrees, M. Mahmoud, and H. Hefny, “Fake account detection in Twitter Based on Minimum Weighted Feature set,” World Academy of Science, Engineering and Technology International Journal of Computer and Information and Engineering, 2016

[7] “List of social networking websites,” *Wikipedia*, 10-Sep-2018. [Online]. Available:https://en.wikipedia.org/wiki/List_of_social_networking_websites [Accessed Oct. 9, 2018]

[8] “List of virtual communities with more than 100 million active users,” Wikipedia, 28-Sep-2018. [Online]. Available: https://bit.ly/1RuUNJA [Accessed Oct. 9, 2018]

[9] “Top 15 Most Popular Social Networking Sites | May 2018,” [Online]. Available: https://bit.ly/1d2L2hr [Accessed Oct. 9, 2018]

[10] “Twitter Usage Statistics,” Google Search Statistics - Internet Live Stats. [Online]. Available: http://www.internetlivestats.com/twitter-statistics/ [Accessed Oct. 9, 2018]

[11] “World Map of Social Networks,” Vincos blog. [Online]. Available: http://vincos.it/world-map-of-social-networks/ [Accessed Oct. 9, 2018]

[12] S. Gurajala, J. S. White, B. Hudson, and J. N. Matthews, “Fake Twitter accounts,”*Proceedings of the 2015 International Conference on Social Media & Society - SMSociety '15*, 2015.

[13] “Twitter's Purge of Fake Accounts Could Lead to Drop in User Numbers,” *Fortune*. [Online]. Available: https://for.tn/2PolCV8 [Accessed Oct. 9, 2018]

[14] A. Considine, “Buying Their Way to Twitter Fame,” *The New York Times*, 22-Aug-2012. [Online]. Available: https://nyti.ms/2C3XEe8 [Accessed Oct. 9, 2018]

[15] Hu, L.-Y., Huang, M.-W., Ke, S.-W., Tsai, C.-F "The distance function effect on k-nearest neighbor classification for medical datasets" SpringerPlus 5 2016

[16] Marr B. "What Is The Difference Between Artificial Intelligence And Machine Learning?" [Internet]. Forbes. Forbes Magazine; (2017). Available from: https://bit.ly/2C4qmeS [Accessed Oct. 9, 2018]

[17] Raschka, S, "Python Machine learning", Packt, Birmingham, 2015, Pp.(3,4,7, 70, 77, 90, 92, 93, 128, 129, 314)

[18] Patel, S. "Chapter 2 : SVM (Support Vector Machine) - Theory" [Internet] 2017, Available from: https://bit.ly/2pFAPFo. [Accessed Oct. 9, 2018]

[19] None, "Support Vector Machines: A Guide for Beginners" [Internet]. Available from: https://bit.ly/1Ti383l [Accessed Oct. 9, 2018]

[20] Donges N. "The Random Forest Algorithm – Towards Data Science" [Internet] Towards Data Science 2018. Available from: https://bit.ly/2u7erIB

[21] Zakka K, "A Complete Guide to K-Nearest-Neighbors with Applications in Python and R.", [Internet], 2016, Available from: https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

[22] Hu, L.-Y., Huang, M.-W., Ke, S.-W., Tsai, C.-F "The distance function effect on k-nearest neighbor classification for medical datasets" SpringerPlus 5

[23] Sutton, RS and Barto, AG, "Reinforcement Learning: An Introduction",A Bradford Book, None, 2017, Pp. 1-2, 42-44

[24] "Understanding the 3 Categories of Machine Learning - AI vs. Machine Learning vs. Data Mining 101 (part 2)," Guavus - Go Decisively, 15-Aug-2018. [Online]. Available: https://bit.ly/2IIZ9zK [Accessed Oct. 9, 2018]

[25] Keen BA. "K-means Clustering in Python" [Internet]. Ben Alex Keen. 2017. Available from: http://benalexkeen.com/k-means-clustering-in-python/

[26] Seif G. "The 5 Clustering Algorithms Data Scientists Need to Know" [Internet] Towards Data Science; 2018. Available from: https://bit.ly/2BomxkA

[27] About Twitter's APIs [Internet]. Twitter. Twitter; Available from: https://help.twitter.com/en/rules-and-policies/twitter-api [Accessed Oct. 9, 2018]

[28] The Fake Project, Dataset, Available from: http://mib.projects.iit.cnr.it/dataset.html [Accessed Oct. 9, 2018]

[29] Cresci S, Pietro RD, Petrocchi M, Spognardi A, Tesconi M "A Fake Follower Story: improving fake accounts detection on Twitter" Technical report 2014

[30] Wei Q, Dunbrack RL. "The Role of Balanced Training and Testing Data Sets for Binary Classifiers in Bioinformatics." PLoS ONE. 2013

[31] None, "Correlation Coefficient: Simple Definition, Formula, Easy Steps" [Internet]. Statistics How To. Available from: https://bit.ly/2CN4IsQ [Accessed Oct. 9, 2018]

[32] Dalinina R. "Introduction to Correlation" [Internet]. DataScience.com. Blog; Available from: https://bit.ly/2yswJ8i [Accessed Oct. 9, 2018]

[33] "scikit-learn: machine learning in Python - scikit-learn 0.16 documentation," *1.4. Support Vector Machines - scikit-learn 0.19.2 documentation*. [Online]. Available: http://scikit-learn.org/ [Accessed Oct. 9, 2018]

[34] "Why, How and When to apply Feature Selection – Towards Data Science," *Towards Data Science*, 31-Jan-2018. [Online]. Available: https://bit.ly/2OFHRsT

[35] VanderPlas J. "In Depth: Principal Component Analysis" [Internet]. Introducing Scikit-Learn | Python Data Science Handbook. Available from: https://bit.ly/2BFPS8f [Accessed Oct. 9, 2018]

[36] S. Li, "Building A Logistic Regression in Python, Step by Step," *Towards Data Science*, 29-Sep-2017. [Online]. Available: https://bit.ly/2C6kIZX

[37] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural Computing and Applications*, vol. 24, no. 1, pp. 175–186, 2013.

[38] Cover TM, Thomas JA (2006) Elements of Information Theory. 2nd edn. Wiley-Interscience, New Jersey

[39] Anaconda, "What is Anaconda?," *Anaconda*, 08-Oct-2018. [Online]. Available: https://www.anaconda.com/what-is-anaconda/ [Accessed Oct. 9, 2018]

[40] None, "F1 score" [Internet]. Wikipedia Wikimedia Foundation, 2018, Available f rom: https://en.wikipedia.org/wiki/F1_score [Accessed Oct. 9, 2018]

[41] "Evaluation of Clustering Algorithms," *Data Clustering: Theory, Algorithms, and Applications*, pp. 299–320, 2007

[42] "sklearn.neighbors.KNeighborsClassifier¶," *1.4. Support Vector Machines - scikit-learn 0.19.2 documentation*. [Online]. Available: https://bit.ly/1MTzw9M

[43] "sklearn.svm.SVC¶," *1.4. Support Vector Machines - scikit-learn 0.19.2 documentation*. [Online]. Available: https://bit.ly/2pKrwoq

[44] "3.2.4.3.1. sklearn.ensemble.RandomForestClassifier¶," *1.4. Support Vector Machines - scikit-learn 0.19.2 documentation*. [Online]. Available: https://bit.ly/2fDrBrA [Accessed Oct. 10, 2018]

[45] Prabhu, "Understanding Hyperparameters and its Optimisation techniques," *Towards Data Science*, 03-Jul-2018. [Online]. Available: https://bit.ly/2PmxEi1 [Accessed Oct. 10, 2018]

[46] None, "Feature scaling" [Internet]. Wikipedia. Wikimedia, Foundation, 2018 Available from: https://en.wikipedia.org/wiki/Feature_scaling

[47] Raschka, S. "About Feature Scaling and Normalization" [Internet] 2014 Available from: http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

[48] Jabbar HK, Khan RZ. "Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning" (Comparative Study). Computer Science, Communication and Instrumentation Devices. 2014

[49] Chih-Wei H., Chih-Chung C., and Chih-Jen L. „A Practical Guide to Support Vector Classification" Department of Computer Science National Taiwan University, Taipei 106, Taiwan

[50] None, "Beyond Accuracy: Precision and Recall" [Internet]. Towards Data Science. Towards Data Science; 2018. Available from: https://bit.ly/2C5qRpd

[51] Glander S. "Dealing with unbalanced data in machine learning" [Internet]. Sitewide ATOM. Available from: https://shiring.github.io/machine_learning/2017/04/02/unbalanced

[52] None, "What is a Confusion Matrix in Machine Learning" [Internet]. Machine Learning Mastery. 2018. Available from: https://machinelearningmastery.com/confusion-matrix-machine-learning

[53] "sklearn.model_selection.GridSearchCV¶," *1.4. Support Vector Machines - scikit-learn 0.19.2 documentation*. [Online]. Available: https://bit.ly/2sIJSXK

[54] Girard J. "What are C and gamma with regards to a support vector machine? [Internet]. Quora, 2018. Available from:https://www.quora.com/What-are-C-and-gamma-with-regards-to-a-support-vector-machine [Accessed Oct. 10, 2018]

[55] Ben-Hur A. Weston J. "A User's Guide to Support Vector Machines" , Part of the Methods in Molecular Biology book series (MIMB, volume 609), 2009 Pp.10

[56] DFteam, "Kernel Functions-Introduction to SVM Kernel & Examples" [Internet]. Machine Learning Tutorials by DF Team DataFlair. 2017. Available from: https://data-flair.training/blogs/svm-kernel-functions [Accessed Oct. 10, 2018]

[57] Raschka S. "How to Select Support Vector Machine Kernels" [Internet] Michigan State University, Available from: https://bit.ly/2IJZD8S

[58] Plisson, F. "Tuning hyperparameters in Random Forest using Grid Search" [Internet] Available from: https://bit.ly/2zZqriv [Accessed Oct. 10, 2018]

[59] None, "sklearn.ensemble.RandomForestClassifier" [Internet]. Documentation. Available from: https://bit.ly/2fDrBrA [Accessed Oct. 9, 2018]

[60] Fraj MB. "In Depth: Parameter tuning for Random Forest" – All things AI – Medium [Internet]. Medium. Augmenting Humanity; 2017. Available from: https://bit.ly/2C5KU6D [Accessed Oct. 9, 2018]

[61] „What-is-the-difference-between-supervised-and-unsupervised-learning-algorithms", Quara [Internet] Available from: https://bit.ly/2b4esU3

[62] Support Vector Machines: Wikipedia [Internet] Available from: https://bit.ly/2Oh8TXZ [Accessed Oct. 10, 2018]

[63] Principal Component Analysis fundamental – [Online]. Available: http://setosa.io/ev/principal-component-analysis/. [Accessed: Sep. 30th 2018]

[64] "Predictive modeling, supervised machine learning, and pattern classification," Dr. Sebastian Raschka, 25-Aug-2014. [Online]. Available: https://sebastianraschka.com/Articles/2014_intro_supervised_learning.html.

ANNEXES

Annex A: Features

Each from the five datasets obtained from the Institute of Informatics and Telematics (IIT) is composed of four comma-separated values (CSV) files - users.csv, tweets.csv, followers.csv and friends.csv. The features in each file and their explanation follow.

Users:

Feature	Definition
id	Unique identifier number for every user
created_at	The UTC date time that the user account was created on Twitter
dataset	The label of the sample. Indicates if the user is fake or real
default_profile	When true, indicates that the user has not altered the theme or background of their user profile
default_profile_image	When true, indicates that the user has not uploaded their own profile image and a default image is used instead
description	The user-defined UTF-8 string describing their account
followers_count	The number of followers this account currently has
fovourites_count	The number of likes each account has marked
friends_count	The number of users this account is following (AKA their “followings”)
geo_enabled	Indicates that the user has enabled the possibility of geotagging their Tweets
lang	The BCP 47 code for the user’s self-declared user interface language
listed_count	The number of public lists that this user is a member of
location	The user-defined location for this account’s profile
name	The name of the user, as they’ve defined it
profile_background_color	The hexadecimal colour chosen by the user for their background
profile_background_image_url	A HTTP-based URL pointing to the background image the user has uploaded for their profile
profile_background_image_url_https	A HTTPS-based URL pointing to the background image the user has uploaded for their profile
profile_background_tile	indicates that the user’s *profile_background_image_url* should be

	tiled when displayed
profile_banner_url	The HTTPS-based URL pointing to the standard web representation of the user's uploaded profile banner
profile_image_url	A HTTP-based URL pointing to the user's profile image
profile_image_url_https	A HTTPS-based URL pointing to the user's profile image
profile_link_color	The hexadecimal colour the user has chosen to display links with in their Twitter UI
profile_sidebar_border_color	The hexadecimal colour the user has chosen to display sidebar borders with in their Twitter UI
profile_sidebar_fill_color	The hexadecimal colour the user has chosen to display sidebar backgrounds with in their Twitter UI
profile_text_color	The hexadecimal colour the user has chosen to display text with in their Twitter UI
profile_use_background_image	indicates the user wants their uploaded background image to be used
protected	Indicates that this user has chosen to protect their Tweets
screen_name	The screen name, handle, or alias that this user identifies themselves with
statuses_count	The number of Tweets (including retweets) issued by the user
time_zone	The time zone where that user created the account
url	A URL provided by the user in association with their profile
utc_offset	
verified	Indicates that the user has a verified account

Followers:

Feature	Definition
source_id	The id of the source profile
target_id	The source profile has been followed by this id´s profile

Friends:

Feature	Definition
source_id	The id of the source profile
target_id	The id of the profile who the source profile is following

Tweets:

Feature	Definition
created_at	UTC time when this Tweet was created
favorite_count	The favorite_count provides the number of times the tweet has been favourited. In the case of a retweet, favorite_count is the favourite count of the source tweet
geo	This deprecated attribute has its coordinates formatted as [lat, long], while all other Tweet geo is formatted as [long, lat].
id	The identification number of the tweet
in_reply_to_screen_name	If the represented Tweet is a reply, this field will contain the screen name of the original Tweet's author.
in_reply_to_status_id	If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's ID.
in_reply_to_user_id	If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's author ID.
num_hashtags	The number of hashtags each tweet contains
num_mentions	The number of users mentioned in each tweet
num_urls	The number of URLS this tweet contains
place	When present, indicates that the tweet is associated (but not necessarily originating from) a Place .
reply_count	Number of times this Tweet has been replied to
retweet_count	This feature provides the number of times the source tweet was retweeted
retweeted_status_id	The identification number of the tweet
source	Utility used to post the Tweet, as an HTML-formatted string
text	The actual UTF-8 text of the status update
truncated	Indicates whether the value of the text parameter was truncated, for example, as a result of a retweet exceeding the original Tweet text length limit of 140 characters.
user_id	The identification number of the profile who wrote this tweet

Annex B: Default hyperparameters

Hyperparameters are the parameters of a classifier or estimator that are not directly learned in the machine learning step from the training data but are optimized separately and in advance. In the following three tables are defined all the parameters of kNN, SVM and RF and their default values defined bz the scikit-learn library.

KNeighborsClassifier			
Parameters	**Explanation**	**Default value**	**Tuning parameter range**
n_neighbors	Number of neighbors	5	1-100
metric	Distance metric to use for the tree	‘minkowski‘	{‘minkowski‘, ‘euclidean‘, ‘chebyshev‘, ‘manhattan‘}
p	Power parameter for the Minkowski metric	2	3-5
weights	Weight function used in prediction. When uniform, all points in each neighborhood are weighted equally	‘uniform’	‘uniform’
algorithm	Algorithm used to compute the nearest neighbor. If ‘auto’, it will attempt to decide the most appropriate algorithm. Other opportunities: ‘ball_tree’, ‘kd_tree’, ‘brute’	‘auto‘	‘auto‘
leaf_size	Leaf size passed to BallTree or KDTree.	30	30
metric_params	Additional keyword arguments for the metric function.	None	None

SupportVectorClassification			
Parameters	**Explanation**	**Default value**	**Tuning parameter range**
C	Penalty parameter C of the error term.	1	{0.001, 0.01, 0.1, 0.362, 1.0, 3.62, 10.0, 31.62, 100.0, 300.0}
kernel	Specifies the kernel type to be used in the algorithm. It must be one of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable.	‘rbf‘	{‘linear‘, ‘rbf‘}
gamma	Kernel coefficient for ‘rbf’.	1/n_features	{0.001, 0.01, 0.1, 0.362, 1.0, 3.62, 10.0, 31.62, 100.0, 300.0}
degree	Degree of the polynomial kernel function (‘poly’). Ignored by all other kernels.	3	3
coef0	Independent term in kernel function. It is only significant in ‘poly’ and ‘sigmoid’.	0.0	0.0
probability	Whether to enable probability estimates.	False	False
shrinking	Whether to use the shrinking heuristic.	True	True
tol	Tolerance for stopping criterion.	1e-3	1e-3
verbose	Enable verbose output.	False	False
max_iter	Hard limit on iterations within solver, or -1 for no limit.	-1	-1
decision_function_shape	Whether to return a one-vs-rest (‘ovr’) decision function as all other classifiers, or the original one-vs-one (‘ovo’) decision function of libsvm	ovr	ovr
random_state	The seed of the pseudo random number generator to use when shuffling the data	None	None

RandomForestClassifier			
Parameters	**Explanation**	**Default value**	**Tuning parameter range**
n_estimatos	The number of trees in the forest.	10	1-100
criterion	The function to measure the quality of a split. Gini impurity is what stays behind “gini”.	‘gini‘	‘gini‘
max_features	The number of features to consider when looking for the best split.	Square root of the total number of features	1-12
min_samples_split	The minimum number of samples required to split an internal node.	2	1-1000
min_samples_leaf	The minimum number of samples required to be at a leaf node	1	1-5
max_features	The number of features to consider when looking for the best split	auto	auto
max_depth	The maximum depth of the tree	None	None
min_weight_ fraction_leaf	The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node.	0.	0.
max_leaf_nodes	Grow trees with max_leaf_nodes in best-first fashion. If None then unlimited number of leaf nodes.	None	None
min_impurity _decrease	A node will be split if this split induces a decrease of the impurity greater than or equal to this value.	0.	0.
bootstrap	Whether bootstrap samples are used when building trees.	True	True
oob_score	Whether to use out-of-bag samples to estimate the generalization accuracy.	False	False
random_state	If int, random_state is the seed used by the random number generator;	None	123